▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 4

PASO A PASO



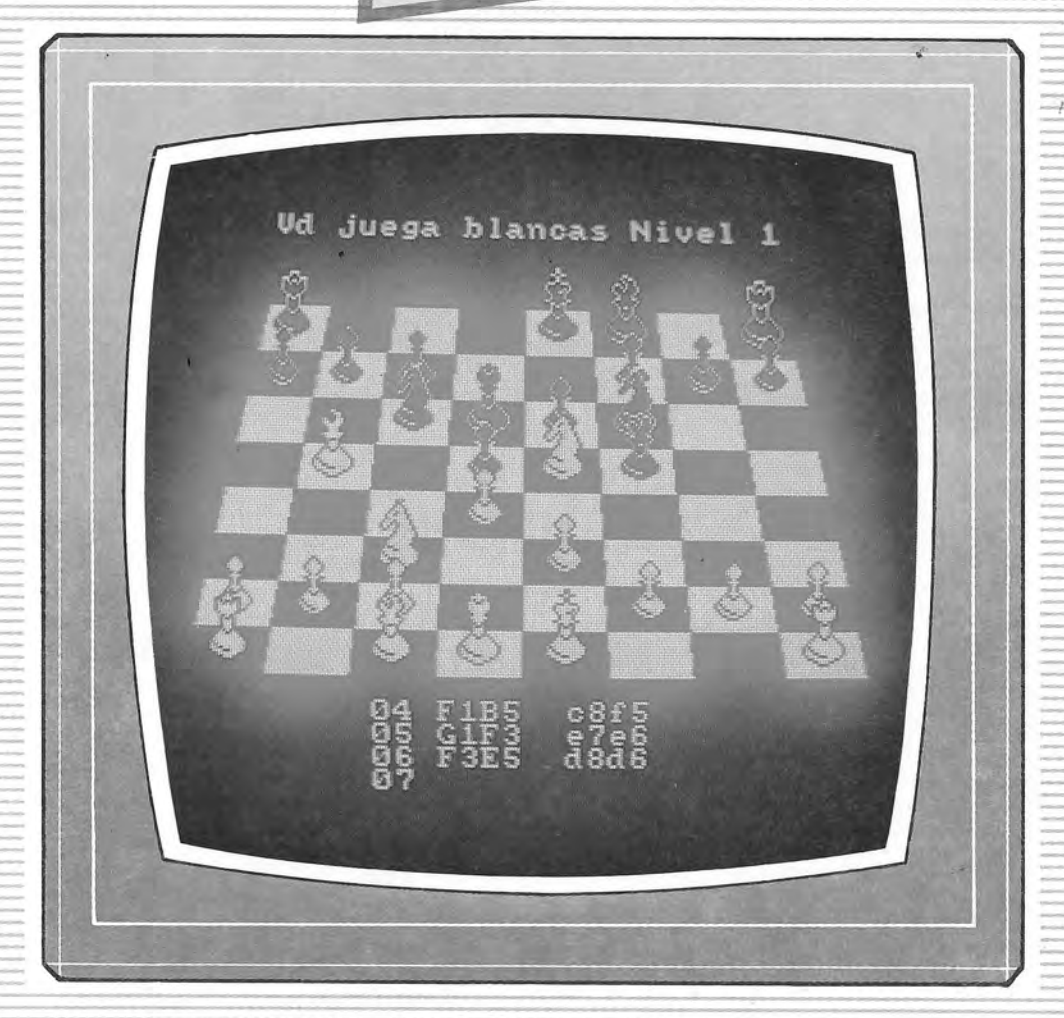
PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Informática y programación 4

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

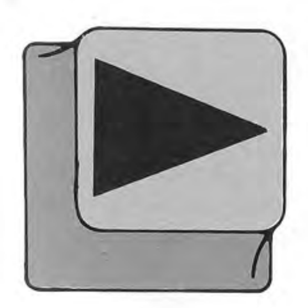
Director-editor:
RICARDO ESPAÑOL CRESPO.
Gerente:
ANTONIO G. CUERPO.
Directora de producción:
MARIA LUISA SUAREZ PEREZ.
Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.
Diseño y maquetación:
BRAVO-LOFISH.
Fotografía:
EQUIPO GALATA.
Dibujos:
JOSE OCHOA

TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas, Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales, Técnico en Informática y colaboradores. Coordinador de Aula de Informática Aplicada (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomuniación y José Luis Todo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: Jesús Bocho, Licenciado en Informática. LOGO: Cristina Manzanero, Licenciada en Informática. APLICACIONES: Fernando Suero, Diplomado en Telecomunicación. OTROS LENGUAJES (Sistemas operativos): Domingo Villaseñor, Diplomado en Informática, y Lenguaje C: Enrique Serrano, Ingeniero en Telecomunicación.

Ediciones Siglo Cultural, S.A.
Dirección, redacción y administración:
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.
Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.
Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.
Distribución en Ecuador: Muñoz Hnos.
Distribución en Perú: DISELPESA.
Distribución en Chile: Alfa Ltda.
Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-072-3
ISBN de la obra: 84-7688-068-7
Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.
Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M. 5.677-1987
Printed in Spain - Impreso en España.
Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.
Abril, 1987.
P.V.P. Canarias: 335,–.



INDICE

INFORMATICA BASICA

ORDENADORES: TIPOS Y TAMAÑOS



Ordenadores, minis y micros

UNO de los puntos más ambiguos en el campo informático se refiere a la clasificación de computadores. Una primera división se podría basar en el tamaño, capacidad y potencia. Atendiendo a estos puntos, los grupos principales serían: **ordenadores, miniordenadores** y **microordenadores.** Sin embargo, profundizando más en el tema aún podríamos añadir ordenadores mucho más potentes como: **mainframes, superordenadores** y **ordenadores especializados.**

En este tema nos vamos a dedicar a estudiar detenidamente cada uno de estos grupos. . .

Ordenadores

Dentro de este tipo se encuentran aquéllos que realizan muchos trabajos al día, es decir, son los utilizados en empresas donde se necesita procesar gran cantidad de datos. Esto supone que dichos ordenadores tengan una serie de características fundamentales:

— Al tener que tratar mucha información, la cantidad de datos con que trabajan será bastante alta; esto requiere unas memorias principales con gran capacidad.

— Por otro lado, como en las empresas normalmente se tienen fuertes restricciones de tiempo, las exigencias en cuanto a rapidez en el proceso de información también son grandes. Por esto es necesaria la utilización de programas de alto nivel cuyas instrucciones son complejas. Este punto influiría también en que las memorias no son sólo de gran capacidad, sino también de gran rapidez.

— Normalmente, la adopción de grandes ordenadores obliga a realizar fuertes inversiones, tanto por lo caros que resultan los equipos, como por las instalaciones que requieren: locales auxiliares amplios y diáfanos, dotados de aire acondicionado, etc.

— En cuanto al equipo humano, no sólo tiene que ser numeroso, sino que, además, debe estar especializado en los lenguajes de explotación y manipulación del equipo.

Miniordenadores

Cuando la complejidad o las características de las aplicaciones no requieren la utilización de equipos tan grandes, se recomienda instalar este otro tipo. Aunque de tamaño reducido, al igual que su costo, sus funciones son las mismas que las de los ordenadores.

— Las memorias, por tanto, también serán más pequeñas; nunca necesitarán almacenar tanta información como en el primer caso.

— Generalmente sólo ejecuta un programa cada vez.

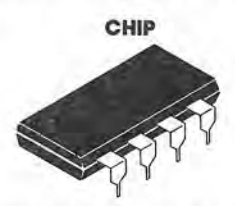
— Esta característica anterior repercute en que el número de usuarios que acceden a la vez será más restringido.

En cuanto a las aplicaciones que suelen realizar están:

— Control de procesos: es capaz de determinar el estado de un proceso y llevar a cabo su corrección, si es preciso, mediante las señales que capta y emite. Como ejemplo, podría ser el control de calidad, inspección material, etc.

— Comunicaciones: en este campo es donde los minis se encuentran en constante desarrollo. Sus aplicaciones típicas son: reserva de billetes, transmisión de mensajes, etc.

— Sistemas de información: aunque estas aplicaciones son más típicas de los ordenadores más grandes, también los minis pueden realizar estas labores típicas de mecanización, como los sistemas comerciales, financieros, de gestión, etc.



Los circuitos integrados en una sola pastilla (CHIP) ha sido determinante para la aparición de los "minis" y "micros".



Microordenadores

El tamaño es el más reducido de toda la clasificación. El descubrimiento del silicio ha hecho que la CPU, que en un principio estaba compuesta por multitud de transistores, quede muy reducida en cuanto a tamaño con la posibilidad de integrar todos sus componentes en una única pastilla. Es muy difícil prever el desarrollo en la reducción de componentes, sin embargo, con los nuevos avances en fotolitografía (grabación por medio de la acción de la luz) se conseguirá una mayor integración de los componentes.

Por tanto, las principales características de los microordenadores son:

— Están compuestos internamente de una única pastilla de CPU, es decir, sólo un procesador es el responsable del funcionamiento de todo el micro.

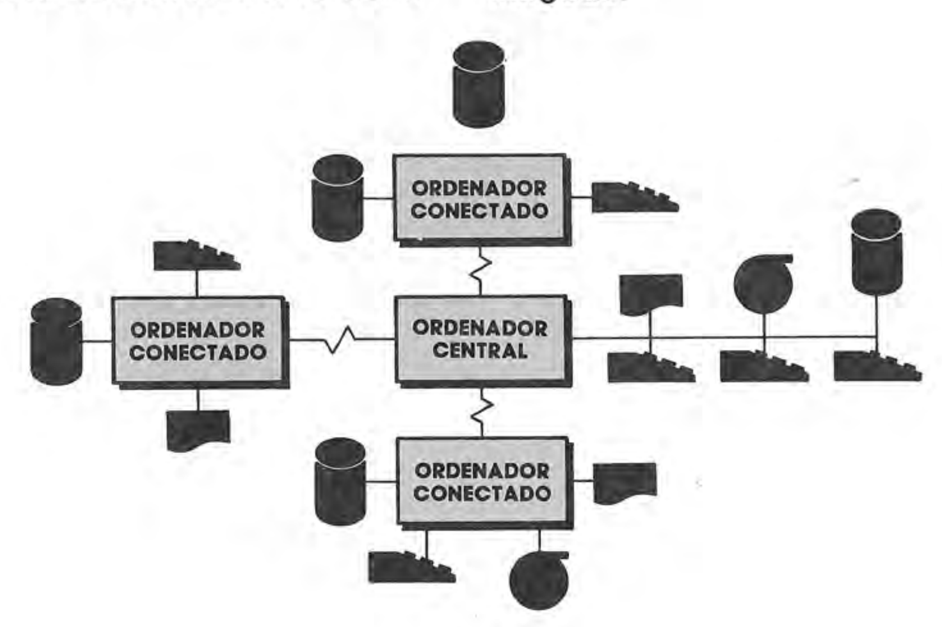
— Las instrucciones utilizadas en los programas también son más sencillas, actuando sobre palabras de 4, 8, o 16 bits.

— El precio es bastante bajo, en comparación con los otros grupos vistos anteriormente; esto ha contribuido a su gran difusión, sobre todo en el campo de la enseñanza.

— Se ha restringido su uso a aplicaciones particulares, como:

— Control de periféricos: pudiendo la CPU quedar libre para ejecutar otras tareas.

— Toma de datos: el micro puede recibir datos de diversas fuentes, los puede tratar según el programa que esté cargado.



Otra forma de poder ampliar las posibilidades de acceso a mayor volumen de información es la conexión de micros mediante las llamadas REDES DE ORDENADORES.



Mainframes

La avidez de información ha hecho que los pequeños usuarios tengan necesidad de comunicarse con los grandes ordenadores y así tener acceso a grandes cantidades de información, que en un pequeño ordenador no sería posible almacenar. Basándose en esta necesidad, las grandes empresas fabricantes de ordenadores han creado los "mainframes", de manera que sus micros y miniordenadores puedan ser conectados a éstos y ampliar así su relativamente pequeña capacidad.

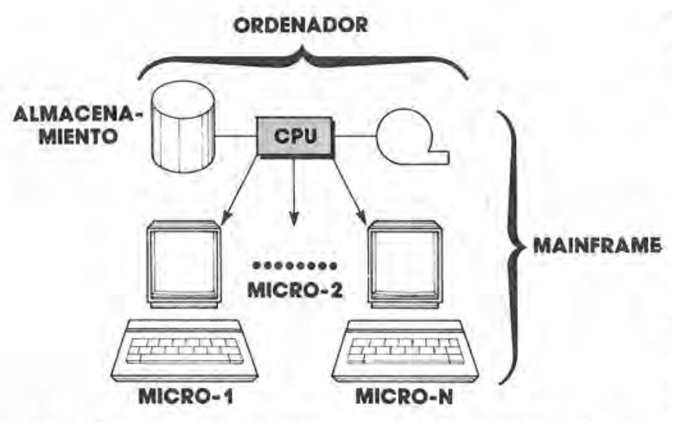
Los principales usuarios de este tipo de relación son los centros de cálculo, que, disponiendo de un ordenador grande, conectan a su alrededor una serie de pequeños ordenadores que juntos crean un potente sistema, pudiendo a su vez funcionar autónomamente. Al estar compuesto por ordenadores de bajo coste este sistema se hace enormemente eficaz.

Características de los mainframes

— Los mainframes se diferencian de los minis y microordenadores en que mientras los minis y micros han sido diseñados para tareas específicas, los mainframes pueden ser utilizados en todo tipo de tareas. Si bien es cierto que los micros y minis son utilizados en la actualidad para todo tipo de tareas.

— Debido al alto coste de los mainframes, su número es bastante reducido; sin embargo, su potencial es mucho mayor y su utilidad supera con mucho a la de los pequeños equipos. A pesar de todo, la ventaja de los pequeños ordenadores en su relación precio-potencia sigue siendo muy superior.

— Con la aparición de los micros y miniordenadores, se han creado una serie de empresas, dedicadas al diseño de software específico para los mainframes, provocándose así una importante diferencia entre unos y otros.



Para poder acceder a la información de un gran sistema desde pequeño equipo se hacen conexiones entre los grandes ordenadores y microordenadores compatibles.



Ordenadores especializados

La gran variedad de posibilidades que ofrece hoy la informática ha hecho que las empresas que deciden informatizar su funcionamiento soliciten la creación de equipos informáticos especialmente diseñados para éstas, de manera que puedan ser satisfechas sus necesidades.

Sería imposible catalogar todos estos equipos, pues su crecimiento es cada vez mayor y abarca prácticamente todos los campos y facetas posibles.

Podemos tomar como ordenadores especializados más importantes, en base a su gran desarrollo: el procesador de imágenes, los sistemas robots y el sistema de diseño mecanizado.

Procesador de imágenes

Este sistema ha sido muy desarrollado; en la actualidad han conseguido mejorar, incluso, la fotografía original. El proceso consiste en convertir la información que proporciona una imagen dada a información digital, para su posterior manipulación con el ordenador. De esta forma podemos mejorar la calidad de una imagen que no esté en condiciones óptimas, si conocemos las razones de su falta de calidad y las rectificamos.

Para la conversión de la imagen analógica en digital, utilizamos un aparato llamado "microdensímetro". Este hace pasar un haz de luz colimado (es decir, cuyos rayos son paralelos) a través de la imagen en un soporte parecido al de la diapositiva fotográfica. La luz que pasa a través de la fotografía es recogida y transformada en impulsos analógicos por medio de un "fotomultiplicador", que genera una señal eléctrica cuya intensidad está en función del brillo de cada punto de imagen. Después, la señal analógica es digitalizada con un cuantizador, que lo transmite al ordenador. Este sistema no

sólo permite mejorar una imagen sino que también podemos describir ésta mediante escritura, lo que permite hacer el proceso contrario, a partir de información escrita podemos obtener una imagen.

Sistemas robot

Los robots actuales tienen su origen en los autómatas que aparecieron en Europa en los siglos XVIII y XIX, aunque en esta época eran considerados como algo "mágico".

Actualmente, puede hablarse claramente de dos tipos de robots. Los primeros son los dependientes de un ordenador central que les facilita la información para desempeñar una determinada función. Estos se utilizan ya en empresas y cadenas de montaje, siendo capaces incluso de tomar decisiones no determinadas "a priori", pero siempre dentro de un medio predeterminado. La movilidad de estos robots está limitada a su medio, es decir, a su campo de trabajo.

El segundo tipo de robots son los llamados autónomos, es decir, robots capaces de determinar el medio en el que se encuentran y desenvolverse en él. Tal es el caso del robot desarrollado en el MIT (Massachusetts Institute of Technology), que posee tres niveles de "inteligencia", correspondientes a tres etapas superpuestas de ordenadores. Un nivel para el reconocimiento del espacio y medio en que se desenvuelve; un segundo nivel para el reconocimiento de obstáculos y un tercer nivel que decide cuál es el camino más factible para evolucionar. Actualmente este tipo de sistemas robots está en vías de desarrollo; sin embargo, su utilización en actividades que resultan peligrosas para el hombre, como puede ser la manipulación de materiales radiactivos, explosivos, etc., está suficientemente probada y justificada.

Sistema de diseño mecanizado

Ha sido creada una serie de ordenadores especializados en el tratamiento de diseños. Estos hacen que el proceso que transcurre desde que se concibe un diseño hasta que se elabora sea acortado enormemente, puesto que el paso intermedio de su plasmación en el papel ha sido automatizado. Un operador, mediante un lápiz óptico, puede mover y corregir en muy poco tiempo todo cuanto quiera.

Actualmente el diseño por ordenador está alcanzando unas cotas realmente altas, ya que su comodidad y rapidez dan lugar a un mayor tiempo para la creación. De esta manera, están surgiendo nuevas facetas en el diseño por ordenador como es el caso de la "animación por ordenador", que se basa en la creación, por medio de complejos programas, de imágenes animadas. Este tipo de diseño está abarcando todo tipo de campos, desde el publicitario, hasta el artístico. Sus grandes posibilidades se deben a que, como se trabaja en un espacio totalmente creado, es a su vez totalmente transformable.

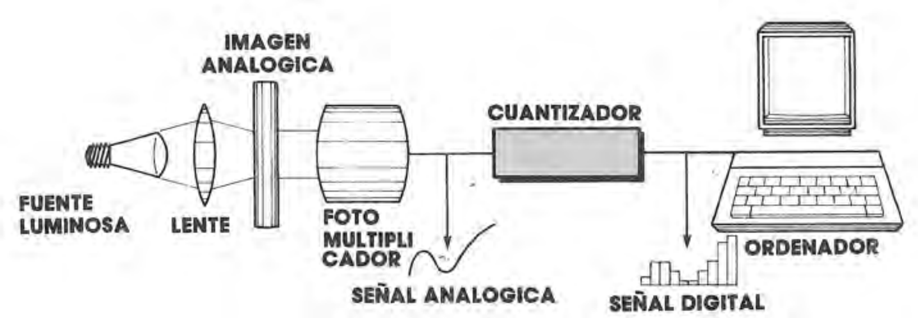


Superordenadores

Considerados como auténticas rarezas, no existen hoy en el mundo más de sesenta superordenadores. Tan sólo algunas, poderosas, entidades sacan partido de estas poderosas máquinas (centros científicos y militares).

De momento, sólo hay dos marcas en el mercado: los CRAY, de Cray Research y los CYBER, de Control Data Corporation; aunque hay que destacar los esfuerzos de la empresa japonesa FUJITSU, que intenta introducirse en el reducido grupo.

Estos ordenadores son los "bichos raros" de la informática, se les conoce con



Pasos necesarios para convertir señales luminosas en señales comprensibles por el ordenador especializado en el proceso de imágenes.

el nombre de "number crunchers" (trituradores de números) y es esto lo que realmente hacen. No existe entre ellos y el hombre un contacto directo (carecen de pantallas terminales); de ahí su difícil comparación con los mainframes, micros y miniordenadores. La comunicación entre éstos y el hombre se realiza a través de uno o varios ordenadores, denominados "front-end". Estos se ocupan de que el superordenador no desperdicie ni un ápice de su tiempo.

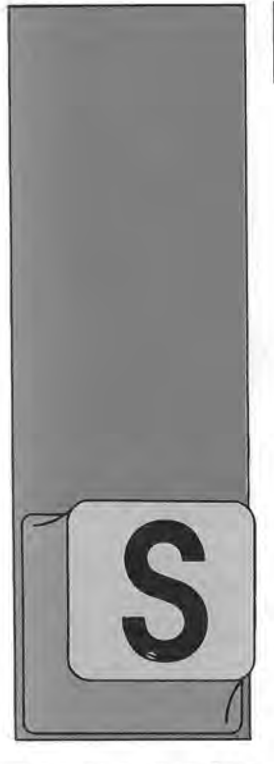
Los superordenadores se emplean a fondo, en la simulación de complicados procesos en los llamados "modelos aritméticos de tiempo real". Algunos institutos meteorológicos utilizan estos superordenadores para calcular la evolución de grandes zonas de bajas presiones. También son utilizados en el diseño de aviones donde se encargan de calcular las turbulencias de aire alrededor del aparato; no obstante, los diseñadores han de conformarse con estos cálculos en porciones, a partir de las cuales hacen un modelo del total y esperan que funcione.

Estas poderosas máquinas realizan entre 20 y 30 millones de operaciones por segundo. Sin embargo, es posible conseguir que la unidad de procesamiento, en su limite, llegue a producir impulsos cortos de 400 millones de operaciones por segundo. Aun así, los científicos estiman que son demasiado lentos. Su gran consumo y mantenimiento hace que estos ordenadores sean exclusivos de unos pocos. Hemos de tener en cuenta que un Cray consume tanta electricidad como un edificio de oficinas de diez pisos y que el calor que produce debe ser disipado mediante una instalación de refrigeración especial a base de freón.

MAQUINA 8088



Unidad Central de Procesos



E suele considerar la «Unidad Central de Procesos» del ordenador (CPU) como una «caja negra», es decir, como una unidad de la que no nos importa cómo está hecha sino cómo ejecutar las instrucciones.

Desde este punto de vista, nos basta realmente con la descripción de las instrucciones con las que prepararemos los programas.

Pero debemos ser conscientes de que la CPU es algo sumamente complejo, que está constituido por unidades más elementales.

Los principales componentes del microprocesador 8088 de Intel que constituye la CPU del IBM PC son:

- Una unidad de cálculo de direcciones de memoria.
- Cuatro «registros de segmento» (llamados DS, ES, SS y CS).
- Un «registro apuntador de instrucciones» (llamado IP).
- Una unidad de preparación de instrucciones, que las lee de la memoria y las prepara en una cola de instrucciones.
- Una unidad de control de ejecución, que toma las instrucciones de la cola y se las pasa a la unidad de ejecución.
- Una unidad de ejecución, que es la que se encarga en último término de ejecutar las instrucciones, y que se compone a su vez de:

— Una «unidad aritmética lógica», que, como su nombre indica, se encarga de realizar las operaciones aritméticas y lógicas.

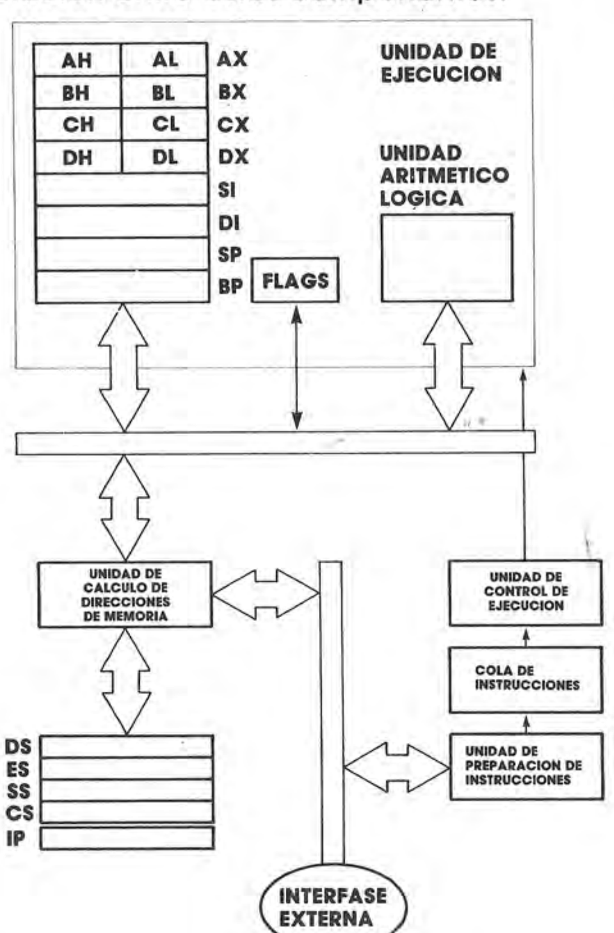
— Cuatro «registros generales» (llamados AX, BX, CX y DX).

— Dos «registros índices» (llamados SI y DI).

— Dos «registros apuntadores» (llamados SP y BP).

— Un «registo de flags» compuesto por una serie de indicadores que influyen en la forma de ejecutarse muchas instrucciones.

El siguiente esquema da una idea de la relación entre estos componentes:





Registros

De los componentes de la CPU citados, a los que más debemos prestar atención es a los registros, los cuales se manejan directamente en las instrucciones y son, por tanto, las piezas claves de la programación.

Lo primero que puede llamar la atención, es que los registros tienen nombres propios (AX, SI, BP, etc.), mientras que en otras CPUs los registros están simplemente numerados. Esto no es un capricho, sino que está relacionado con una característica importante del 8088: «Los registros desempeñan misiones específicas en determinadas instrucciones», o dicho de otra forma: «muchas de las instrucciones del 8088 se ejecutan basándose en el contenido de determinados registros». Los nombres asignados están relacionados con los papeles desempeñados en la ejecución de las instrucciones.

La afirmación anterior quedará más clara con el siguiente ejemplo: la instrucción REPZ STOSW que apareció cuando «des-ensamblamos» el programa MAQUINA (ver tomo primero), significa: «copiar una información repetidamente sobre una zona de memoria». Pues bien, en dicha instrucción se presupone que el registro AX contiene la información a copiar, que el registro CX es el contador que contiene el número de copias a realizar y que los registros ES y DI definen la dirección de comienzo de la zona de memoria deseada.

Hay muchas instrucciones que, como ésta, operan con registros concretos, hay otras que se pueden usar con cualquier registro, y por último, hay otro grupo de instrucciones que se pueden usar con unos sí y con otros no.

Pero pasemos a definir las características de los registros. Están constituidos por 16 bits, que se identifican con los números 0 al 15. El bit 0, que es el menos significativo, se representa en el lado derecho del registro, y el bit número 15, que es el más significativo, queda, por tanto, en el extremo izquierdo.

Para verlo más claro, pongamos un ejemplo en el sistema decimal de numeración: en el número 7618, la cifra del extremo derecho (el 8) es la menos significativa (representa 8 unidades) y la cifra del extremo izquierdo (el 7) es la más significativa (ya que representa 7 millares).

De igual forma, en el número binario de 16 cifras: 1100110011001100, la cifra del extremo derecho, llamado bit 0 (que en el ejemplo es un 0), es la menos significativa (representa 0 unidades), y la cifra del extremo izquierdo, llamada bit 15 (que en el ejemplo es un 1), es la más significativa (ya que representa 1 × 32768).

Los cuatro registros generales: AX, BX, CX y DX, están divididos en 2 registros mitades (que podemos llamar semirregistros) de 1 byte cada uno. Una mitad coincide con los bits 0 al 7, que son los bits menos significativos, y recibe el nombre de mitad «L» (inicial de la palabra inglesa «low», que quiere decir «bajo»). La otra mitad coincide con los bits 8 a 15, que son los bits más significativos, y recibe el nombre de mitad «H» (inicial de la palabra inglesa «high», que quiere decir «alto»).

A la información que se guarda en un registro se le llama «palabra» y equivale a 2 bytes. Pues bien, un registro como el AX se puede utilizar para almacenar «una palabra», referenciándolo con el nombre AX, o se puede utilizar para almacenar «dos bytes», referenciándolo con los nombres AH (para el byte más significativo) y AL (para el menos). Lo mismo le ocurre a los registros BX, CX y DX que se pueden referenciar por estos nombres y por el nombre de sus mitades, BH y BL del primero, CH y CL del segundo, y DH y DL del tercero. Todo esto se resume en la figura siguiente.

Registro																
semi-registro								semi-registro								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bits

	byte más significativo	byte menos significativo
AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL



Conveniencia de la notación hexadecimal

En un semirregistro o byte de memoria se pueden dar 256 configuraciones de bits diferentes (2 elevado a 8), mientras que en un registro o en una palabra de memoria el número de configuraciones de bits es de 65536 (2 elevado a 16).

Estos números, aparentemente tan extraños, aparecen porque la notación decimal que usamos no tiene relación directa con la notación binaria que usa la máquina. Sin embargo, la notación hexadecimal sí tiene una relación directa con la binaria. En efecto, con cuatro dígitos binarios (4 bits) se pueden distinguir 16 estados diferentes que pueden ser representados con una única cifra hexadecimal, del 0 a la F, realizando la siguiente correspondencia:

bits		bits	
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Como en las unidades de memoria usadas (bytes y palabras) el número de bits es múltiplo de 4, tenemos en la notación hexadecimal una forma cómoda y concisa de representar la información manejada por el ordenador.

Los 256 estados posibles asociados a los 8 bits de un byte se pueden representar con los 256 números hexadecimales de dos cifras. La configuración de bits 0000 0000 se representa por el hexadecimal 00 (decimal 0) y la configuración 1111 1111 se representa por el hexadecimal FF (decimal 255).

De la misma forma los 65536 estados diferentes en que pueden encontrarse los 16 bits de un registro, se pueden representar con los 65536 números hexadecimales de 4 cifras. La configuración de bits 0000 0000 0000 0000 se representa por el hexadecimal 0000 (decimal 0), mientras que la configuración 1111 1111 1111 1111 se representa por el hexadecimal FF (decimal 255).



Direccionamiento de la memoria

El microprocesador 8088 puede manejar 1 Megabyte de memoria, es decir, aproximadamente 1 millón de bytes. (El número exacto es 1024 × 1024 = 1048576 bytes). Cada byte de memoria tiene asignado un número secuencial que va desde el 0 (0 hexadecimal) al 1048575 (FFFFF hexadecimal), que es su dirección de memoria (dirección efectiva), con la cual puede ser referenciado de forma inequívoca. La dirección de una palabra de memoria es la dirección del primero de sus dos bytes, que, curiosamente, es el menos significativo de los dos (ya explicaremos esto con más detalle en otro lugar).

Para referenciar todas esas direcciones hacen falta 20 bits, mientras que acabamos de ver que los registros tienen 16 bits y sólo pueden contener 65536 valores diferentes. De esto se deduce que un único registro no puede servir para definir todas las direcciones de memoria que puede usar el microprocesador.

Efectivamente, el 8088 usa siempre dos registros para calcular las direccciones de memoria y tiene un procedimiento particular de cálculo que vamos a ver a continuación. Los dos registros usados se denominan «registro de segmento» y «registro de desplazamiento». Las direcciones efectivas de memoria las calcula el 8088 multiplicando por 16 el número contenido en el «registro de segmento» y sumándole el contenido del «registro de desplazamiento».

Hay que tener en cuenta que multiplicar por 16 en hexadecimal es igual de sencillo que multiplicar por 10 en decimal, es decir, basta con añadir un 0 a la derecha.

Veamos un ejemplo. Si el «registro de segmento» contiene el número hexadecimal 1C7B y el «registro de desplazamiento» contiene 96F8, la dirección efectiva resulta ser 25EA8 (ya que 25EA8 = 1C7B0 + 96F8). La operación se representa en la siguiente figura:

	Registro de Segmento																			
bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
binario	0	0	0	1	1	1	0	0	0	1	1	1	1	0	1	1				
hexadecimal	1				C				7				B				0			

+

					Registro de Desplazamiento															
bits					15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
binario					1	0	0	1	0	1	1	0	1	1	1	1	1	0	0	0
hexadecimal					9				6				F				B			

	Dirección Efectiva																			
bits	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
binario	0	0	1	0	0	1	0	1	1	1	1	0	1	0	1	0	1	0	0	0
hexadecimal	2				5				E				A				8			

Pero no todos los registros pueden hacer el papel de «registros de segmento» y «registros de desplazamiento», sino que hay unas cuantas opciones posibles. Como «registros de segmento» actúan los registros CS, SS, DS y ES. Como «registros de desplazamiento» pueden actuar IP, SP, BP, BX, SI y DI.

Aunque pueden utilizarse muchas parejas de registros válidas, hay cuatro parejas que son típicas, porque direccionan las cuatro áreas de memoria que maneja el 8088. Las áreas y las parejas de registro que las direccionan son las siguientes:

- Area de código (o área de programa). Direccionada por los registros CS e IP.

CS son las iniciales de las palabras inglesas «Code Segment» (que significa «segmento de código»), e IP son iniciales de «Instruction Pointer» (que significa «apuntador de instrucciones»). Estos dos registros direccionan en todo momento la posición de memoria donde se encuentra la siguiente instrucción a ejecutar.

- Area de datos. Direccionada por los registros DS y SI.

DS son las iniciales inglesas de «Data Segment» (que significa «segmento de datos»), y SI son las iniciales de «Source Index» (que significa «índice del origen»). Estos registros definen el área de memoria de donde leen determinadas instrucciones. Por ejemplo: la instrucción LODSW lee en el registro AX la palabra de memoria apuntada por DS y SI.

- Area de datos «extra». Direccionada por los registros ES y DI.

ES son las iniciales de «Extra Segment» (que significa «segmento extra»), y DI son las iniciales del «Destination Index» (que significa «índice del destino»). Estos registros sirven para definir el área de memoria sobre la que escriben determinadas instrucciones. Por ejemplo: la instrucción STOSW escribe el contenido del registro AX en la palabra de memoria apuntada por ES y DI.

- Area de la pila de ejecución. Direccionada por los registros SS y SP.

SS son las iniciales de «Stack Segment» (que significa «segmento de la pila»), y SP de «Stack Pointer» (que significa «apuntador de la pila»). Estos registros sirven para direccionar la pila de ejecución o STACK. El concepto de «pila de ejecución» se explicará con detalle más adelante.

Estas cuatro áreas pueden ser realmente cuatro áreas diferenciadas de la memoria, o pueden solaparse. A continuación vamos a ver un programa ejemplo donde ocurre esto con el «Area de datos» y el «Area de datos extra». Ambas áreas se hacen coincidir, en este caso, con la zona de memoria que contiene la información que se representa en la pantalla (que recibe el nombre de «buffer de pantalla»).



Programa ejemplo

Este programa lo vamos a escribir usando también el Programa DEBUG. El método normal de escribir programas en lenguaje máquina (lenguaje ensamblador) lo veremos más adelante cuando se expliquen las nociones básicas de «edición», «ensamblaje» y «montaje».

En este caso vamos a utilizar el mandato A del DEBUG. Esta letra es la inicial de «assemble», que quiere decir «ensamblar», es decir, obtener el código que entiende la máquina (código máquina) a partir de unas instrucciones escritas en un lenguaje entendible por el hombre («lenguaje simbólico»).

Al escribir «a 100» y pulsar Enter, el DEBUG va escribiendo sobre la pantalla la dirección sobre la que se va guardando el código máquina en la forma CS:IP, es decir, un número hexadecimal de cuatro cifras que es el «registro de segmento» seguido de dos puntos y de otro número hexadecimal que es el «apuntador de instrucciones». A partir de ese momento, nosotros escribiremos las instrucciones (que se listan en minúsculas más abajo) y pulsaremos Enter al final de cada una. El mando A se termina al pulsar Enter sin haber definido ninguna instrucción. El resto del listado corresponde a mandatos que se explicaron paso a paso en el tomo primero.

Con esto realizaremos el proceso inverso al ejemplo del programa MAQUINA, donde se escribió directamente el código máquina y se pidió al DEBUG que generase el código en lenguaje simbólico.

A continuación se reproduce lo que aparece sobre la pantalla durante el proceso de escritura y comprobación de este programa. La comprobación mediante la instrucción «u 100 11e» es importante, ya que, cuando se programa en lenguaje máquina, la mayoría de los errores no pueden ser detectados «a priori» y ocasionan frecuentemente «el cuelgue» de la máquina. Es decir, la pérdida del control de la ejecución por el sistema operativo, por lo que la única solución es volver a cargarlo mediante un nuevo «IPL» (sigla de «Initial Program Load» que significa «carga del programa iniciador»).

Hay que tener en cuenta que los que deseen ejecutar el programa en la pantalla de color deben cambiar la instrucción número cuatro, que es «mov ax,b000» por la instrucción «mov ax,b800». El cambio se debe a que cada tipo de pantalla tiene su buffer en una dirección de memoria diferente.

```
A>DEBUG
-
-a 100
0C47:0100 push ds
0C47:0101 mov  ax,0
0C47:0104 push ax
0C47:0105 mov  ax,b000
0C47:0108 mov  ds,ax
0C47:010A mov  es,ax
0C47:010C mov  cx,f0
0C47:010F push cx
0C47:0110 mov  cx,7d0
0C47:0113 mov  si,2
0C47:0116 mov  di,0
0C47:0119 repz
0C47:011A movsw
0C47:011B pop  cx
0C47:011C loop 10f
0C47:011E retf
0C47:011F

-u 100 11e
0C47:0100 1E            PUSH   DS
0C47:0101 B80000        MOV    AX,0000
0C47:0104 50            PUSH   AX
0C47:0105 B800B0        MOV    AX,B000
0C47:0108 8ED8          MOV    DS,AX
0C47:010A 8EC0          MOV    ES,AX
0C47:010C B9F000        MOV    CX,00F0
0C47:010F 51            PUSH   CX
0C47:0110 B9D007        MOV    CX,07D0
0C47:0113 BE0200        MOV    SI,0002
0C47:0116 BF0000        MOV    DI,0000
0C47:0119 F3            REPZ
0C47:011A A5            MOVSW
0C47:011B 59            POP    CX
0C47:011C E2F1          LOOP   010F
0C47:011E CB            RETF

-n simbolo.com
-r cx
CX 0000
:1f

-w
Writing 001F bytes

-q

A>
```

Cuando termina el programa DEBUG tenemos un nuevo fichero llamado SIMBOLO. COM, que contiene 31 bytes con nuestro programa. Al ejecutarlo podemos ver cómo el contenido de la pantalla se desplaza horizontalmente, de forma que se pierde por la izquierda y aparece por la derecha varias veces. Para observar bien el efecto se recomienda ejecutar el programa SIMBOLO cuando la pantalla esté ocupada.

PROGRAMAS

PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS



Programa: vocabulario francés



L primer programa de este cuarto tomo nos permitirá poner al día nuestros conocimientos del idioma francés. El programa nos irá preguntando una serie de palabras en español que nosotros tendremos que traducir al francés. Para hacer más fácil todo, el programa divide las palabras, según su dificultad, en cuatro niveles distintos. Las instrucciones necesarias para utilizar el programa se dan en el mismo.

```
   1 REM ***********************
   2 REM * VACABULARIO FRANCES *
   3 REM ***********************
   4 REM
   5 REM ***********************
   6 REM * POR OSCAR MORALES   *
   7 REM ***********************
   8 REM *(c)Ed.Siglo Cultural *
   9 REM *(c) 1987             *
  10 REM ***********************
  11 REM
  50 LET o=0: LET w=0: LET x=0
 100 PRINT  INVERSE 1;"VOCALES I
GUALES EN AMBOS IDIOMAS": PRINT
 110 PRINT "  Espanol
 Frances": PRINT
 120 PRINT "       a
    a ": PRINT
 130 PRINT "       i
    i": PRINT
 140 PRINT "       o
o,au,eau": PRINT
 150 PRINT "       e
 ai,ei": PRINT
 160 PRINT "       u
  ou"
 170 PRINT "-------------------
------------"
 180 PRINT  INVERSE 1;"VOCALES N
O EXISTENTES EN ESPANOL": PRINT
 190 PRINT "La  "; INVERSE 1;"u"
; INVERSE 0;" en  frances se p
ronunciaponiendo la boca para de
cir la "; INVERSE 1;"u"; INVERSE
 0;"espanola y diciendo i. La ";
 INVERSE 1;"e"; INVERSE 0;" fran
cesa se pronuncia poniendo la bo
ca para decir "; INVERSE 1;"o";
INVERSE 0;",diciendo "; INVERSE
1;"e"; INVERSE 0;"."
 200 PRINT #0; FLASH 1;"
PULSA UNA TECLA.          "
 210 IF INKEY$="" THEN  GO TO 21
0
 211 CLS : PRINT  INVERSE 1;"
         A V I S O           ---
-----------------------------":
PRINT '’" Se avisa que para los
sustanti-": PRINT "vos hace falt
a poner el  articu-": PRINT "lo.
": PRINT '" Si necesita un apost
ofe utilice el guion.": PRINT '"
 Los nombres propios  han de  ir
con mayuscula al principio."
 212 PRINT : PRINT : PRINT "
        SUERTE"
 213 PRINT : PRINT : PRINT : PRI
NT  FLASH 1;"         PULSA UNA T
ECLA.         "
 214 IF INKEY$="" THEN  GO TO 21
4
 215 CLS : INPUT "QUE NIVEL DESE
AS (1-4)";L: IF L<>1 AND L<>2 AN
D L<>3 AND l<>4 THEN  GO TO 215
 216 PRINT AT 10,8; FLASH 1;"ESP
ERA UN MOMENTO"
 217 IF L=1 THEN  LET d=60
 218 IF l=2 THEN  LET d=145
 219 IF l=3 THEN  LET d=206
 220 IF l=4 THEN  LET d=267
 230 DIM a$(d,20): DIM d$(d,20):
 LET q=0
 240 RESTORE 300
 249 LET a=0
 250 FOR i=1 TO d
 254 READ b$: LET a$(i)=b$
 255 NEXT i
 260 LET c$="": RESTORE 400
 270 FOR b=1 TO d
```

```
274 READ c$: LET d$(b)=c$
275 NEXT b
300  DATA "mesa","coche","libro
","ventana","lapiz","palabra","p
erro","silla","puerta","pared","
cabeza","casa","cigarro","lampar
a","armario","comida","agua","ca
ma","suelo","techo","cielo","nub
e","sol","lluvia","habitacion","
cuarto de estar","cuarto de bano
","cocina","salon","pasillo","bo
lsa","pan","leche","jarron","zap
ato","bombilla","arbol","calle",
"taza","cafe","ciudad","cerveza"
,"cana de cerveza","pastel","vin
o","hora","campo","semana","dia"
,"mes","ano","siglo","boligrafo"
,"goma","sacerdote","alambre","p
iel","luna","violin","hospital"
320 DATA "alfombra","curso","Eu
ropa","Espana","Francia","idioma
","hombre","mujer","nino","nina"
,"abuelo","abuela","madre","padr
e","suegra","suegro","primo","pr
ima","sobrino","sobrina","hijo",
"hija","cunado","cunada","horno"
,"grifo","vajilla","plato","tene
dor","cuchillo","cuchara","ojo",
"ojos","nariz","boca","cara","fr
ente","espalda","hombros","piern
a","mano","pie","brazo","animal"
,"persona","pajaro","paloma","ce
rdo","vaca","cordero","carro","r
amo de flores","cesta","tinta","
pantalla","tintero","cenicero","
tiza","largo","ancho","estrecho"
,"corto","alto","bajo","rubio","
moreno","pelirrojo","fuerte","de
bil","abierto","cerrado","papel"
,"reloj","blanco","negro","verde
","rojo","azul","amarillo"
330 DATA "rosa","naranja","marr
on","gris","violeta","fecha","ed
ad","noche","tarde","manana","es
trella","palido","cuadro","cuadr
ado","triangulo","vida","ser","e
s","el es","el tiene","yo tengo"
,"yo soy","yo voy","el viene","g
uante","pantalon","camisa","jers
ey","chaqueta","botas","sombrero
","calcetin","chaleco","necesita
r","prenda interior","ropa de ve
stir","ropa","abrigo","botella",
"gorra","nombre","cual es?","mi"
,"tu","su","nuestro","vuestro","
fuego","dinero","ayudar","amar",
"cortina","cruz","iglesia","medi
co","policia","bombero","tienda"
,"vela","tijera","vaso","corazon
","pelo","cuaderno","profesor","
circulo","el primero"
350 DATA "cajon","sillon","un",
"una","dos","tres","cuatro","cin
co","seis","siete","ocho","nueve
","diez","muneca","telefono","ba
rba","veneno","pez","sombra","di
ente","lleno","vacio","grande","
pequeno","piedra","roca","rio","
mar","aplauso","campana","golpe"
,"encendedor","radiador","chimen
ea","desayuno","cena","juventud"
,"lago","lazo","madera","hierro"
,"lana","cemento","algodon","lad
rillo","verdad","mentira","es ve
rdad!","rama","peso","quizas","r
azon","asiento","sobre","sello",
"subir","bajar","escalera","ladr
on","sonrisa","azucar"
400 DATA "la table","la voiture
","le livre","la fenetre","le cr
ayon","le mot","le chien","la ch
aise","la porte","le mur","la te
te","la maison","la cigarrette",
"la lampe","l-armoire","la nourr
iture","l-eau","le lit","le sol"
,"le plafond","le ciel","la nuag
e","le soleil","la pluit","la ch
ambre","la salle a manger","la s
alle de bain","la cuisine","le s
alon","le couloir","le sac","le
pain","le lait","le vase","la ch
aussure","l-ampoule","l-arbre","
la rue","la tasse","le cafe","la
 ville","la biere","le demi","le
 gateau","le vin","l-heure","le
champ","la semaine","le jour","l
e mois","l-an","le siecle","le s
tylo","la gomme","le cure","le f
il de fer","la peau","la lune","
le violon","l-hopital"
420 DATA "le tapis","le cours",
"l-Europe","l-Espagne","la Franc
e","la langue","l-homme","la fem
me","l-enfant","la fille","le gr
and-pere","la grand-mere","la me
re","le pere","la belle-mere","l
e beau-pere","le cousin","la cou
sine","le neveu","la niece","le
fis","la fille","le beau-frere",
"la belle-soeur","le four","le r
obinet","la vaisselle","l-assiet
te","la fourchette","le couteau"
,"la cuillere","l-oeil","les yeu
x","le nez","la bouche","la figu
re","le front","le dos","l-epaul
e","la jambe","la main","le pied
","le bras","l-animal","la perso
nne","l-oiseau","le pingeon","le
 porc","la vache","l-agneau","le
 chariot","le bouquet","le panie
r","l-encre","l-ecran","l-encrie
r","le sandrier","la craie"
430 DATA "long","large","etroit
","court","haut","bas","blond","
brun","roux","fort","faible","ou
vert","ferme","le papier","la mo
ntre","blanc","noir","vert","rou
ge","bleu","jaune","la rose","or
ange","marron","gris","violet","
la date","l-age"
440 DATA "la nuit","le soir","l
e matin","l-etoile","pale","le t
ableau","le carre","le triangle"
,"la vie","etre","c-est","il est
","il a","j-ai","je suis","je va
is","il viens","le gant","le pan
talon","la chemise","le chandail
```

```
","la jaquette","la botte","le c
hapeau","le chaussette","le gile
t","avoir besoin","le sous-vetem
ent","le vetement","le linge","l
e pardessus","la bouteille","la
casquette","le nom","quel est?",
"mon","ton","son","notre","votre
","le feu","l-argent","aider","a
imer","le rideau","la croix","l-
eglise","le medesin","la police"
,"le pompier","le magasin","la b
ougie","le ciseau","le verre","l
e coeur","le cheveu","le cahier"
,"le professeur","le cercle","le
 premiere"

450 DATA "le tiroir","le fauteu
il","un","une","dos","trois","qu
atre","cinq","six","sept","huit"
,"neuf","dix","la poupee","le te
lephone","la barbe","le poison",
"le poisson","l-ombre","le dent"
,"plein","vide","grand","petit",
"la pierre","le rocher","la fleu
ve","la mer","l-applaudissement"
,"la cloche","le coup","le briqu
et","le radiateur","la cheminee"
,"le petit dejeuner","le souper"
,"la jeunesse","le lac","le noeu
d","le bois","le fer","la laine"
,"le ciment","le coton","la briq
ue","la verite","la mensonge","c
-est vrai!","la brache","le poid
s","peut-etre","la raison","le s
iege","l-enveloppe","le timbre",
"monter","descendre","l-escalier
","le voleur","la sourise","le s
ucre"

900 RANDOMIZE
1000 IF d<=0 THEN  PRINT "HAS AC
ABADO CON EL NIVEL ";L: PRINT "T
U PORCENTAGE DE ACIERTOS ES DEL
";100-((X*100)/O);" %": STOP
1005 CLS : PRINT "QUIERES QUE TE
 PREGUNTE?"
1006 PRINT : PRINT : PRINT
1010 LET x$=INKEY$
1020 IF x$="s" OR X$="S" THEN  G
O TO 1100
1030 IF x$="n" OR X$="N" THEN  C
LS : PRINT "TU PORCENTAGE DE ACI
ERTOS ES DEL ";100-((X*100)/(O+.
00001));" %": STOP
1040 GO TO 1010
1100 LET r=INT (RND*d)+1: IF a$(
r)="                    " THEN
GO TO 1100
1200 PRINT "LA PALABRA ES ";a$(r
)
1210 INPUT "CUAL ES SU TRADUCCIO
N?";r$: LET O=O+1
1220 FOR h=LEN (r$) TO 20: LET r
$=r$+" ": IF r$=d$(r) THEN  LET
h=20: LET q=1
1225 NEXT h: IF q=1 THEN  PRINT
"CORRECTO": LET Q=0: LET w=0: LE
T W=0: GO SUB 3000: GO TO 900
1230 PRINT '' FLASH 1;"NO ES COR
RECTO": PRINT : PRINT "INTENTALO
 DE NUEVO": LET W=W+1: LET X=X+1
: PAUSE 200: CLS
1235 IF W>=3 THEN  PRINT "LA TRA
DUCCION ERA ";D$(R): LET w=0: PA
USE 200: GO SUB 3000: GO TO 900
1240 GO TO 1200
3000 IF r=d THEN  LET a$(r)="":
LET d$(r)="": LET d=d-1: RETURN
3005 FOR g=r TO d-1
3010 LET a$(g)=a$(g+1): LET d$(g
)=d$(g+1): LET a$(g+1)="": LET d
$(g+1)=""
3020 NEXT g
3030 LET d=d-1: RETURN
```

El programa se ha realizado en un SPECTRUM. Para que el programa funcione en los demás ordenadores, hay que realizar algunos cambios. Estos, según el ordenador, se especifican a continuación.

COMMODORE:

60 PRINT CHR$(147)
100 PRINT "VOCALES IGUALES EN AMBOS IDIOMAS":PRINT
180 PRINT "VOCALES NO EXISTENTES EN ESPAÑOL".PRINT
190 PRINT "La 'U' en francés se pronuncia poniendo la boca para decir la 'U' española y diciendo 'I'. La 'E' francesa se pronuncia poniendo la boca para decir 'O', diciendo 'E'."
200 POKE 214,21:POKE 211,0:PRINT "PULSA UNA TECLA"
210 GET A$:IF A$="" THEN GOTO 210
211 PRINT CHR$(147):PRINT "AVISO":PRINT "-----":PRINT:PRINT "Se avisa que para los sustantivos hace falta poner el artículo.":PRINT:PRINT "Si necesita un apóstrofo utilice el guión.":PRINT:PRINT "Los nombres propios han de ir con mayúscua al principio."
213 PRINT:PRINT:PRINT "PULSA UNA TECLA"
214 GET A$:IF A$="" THEN GOTO 214
215 PRINT CHR$(147):INPUT "QUE NIVEL DESEAS (1-4)"; I: IF L<>1 AND L<>2 AND L<>3 AND L<>4 THEN GOTO 215
216 PRINT:PRINT:PRINT "ESPERA UN MOMENTO"
230 DIM A$(D):DIM D$(D)
900 LET A=RND(-TI)
1005 PRINT CHR$(147);"QUIERES QUE TE PREGUNTE?"
1010 GET X$
1030 IF x$="n" OR X$"="N" THEN PRINT CHR$(147);"TU PORCENTAJE DE ACIERTOS

```
ES DEL ";100-((X*100)//(0+.00001));
" %":END
1100 LET R=INT(RND(0)*D)+1:IF A$(R)=" "
THEN GOTO 1100
1230 PRINT "NO ES CORRECTO":PRINT:PRINT
"INTENTALO DE NUEVO":LET W=W+1:LET
X=X+1:FOR G=1 TO 200:NEXT G:PRINT
CHR$(147)
```

IBM:

```
60 CLS
100 PRINT "VOCALES IGUALES EN AMBOS
IDIOMAS":PRINT
180 PRINT "VOCALES NO EXISTENTES EN ES-
PAÑOL":PRINT
190 PRINT "La 'U' en francés se pronun-
cia poniendo la boca para decir la 'U' es-
pañola y diciendo 'I'. La 'E' francesa se
pronuncia poniendo la boca para decir
'O', diciendo 'E'."
200 LOCATE 21,1:PRINT "PULSA UNA TE-
CLA"
211 CLS:PRINT ''AVISO'':PRINT
"-----":PRINT:PRINT "Se avisa que para los
sustantivos hace falta poner el artícu-
lo.''PRINT:PRINT ''Si' necesita un apóstrofo
utilice el guión."PRINT:PRINT "Los nombres
propios han de ir con mayúscula al prin-
cipio."
213 PRINT:PRINT:PRINT "PULSA UNA TECLA"
216 PRINT "ESPERA UN MOMENTO"
1230 PRINT "NO ES CORRECTO":PRINT:PRINT
"INTENTALO DE NUEVO":LET W=W+1:LET
X=X+1:FOR G=1 TO 200:NEXT G:CLS
```

MSX:

Las variaciones que hay que realizar para que el programa funcione en este ordenador son las mismas que para el IBM, excepto la sentencia LOCATE de la línea 200, que en vez de ser LOCATE 21,1 ha de ser LOCATE 1,21. También hay que cambiar la línea 900 por:

```
900 LET A=RND(-TIME)
```

AMSTRAD:

Para este ordenador hay que hacer los mismos cambios que se dan para el IBM, pero hay que cambiar la sentencia LOCATE de la línea 200 por LOCATE 1,21.



Notas al programa

Este programa tiene almacenadas en total 267 palabras. Al principio, éstas





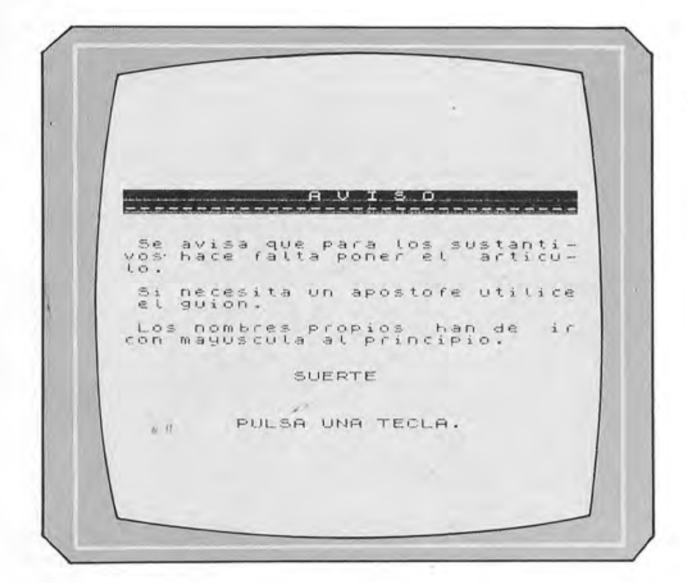
El programa da una pequeña introducción sobre los tipos de la lengua francesa.

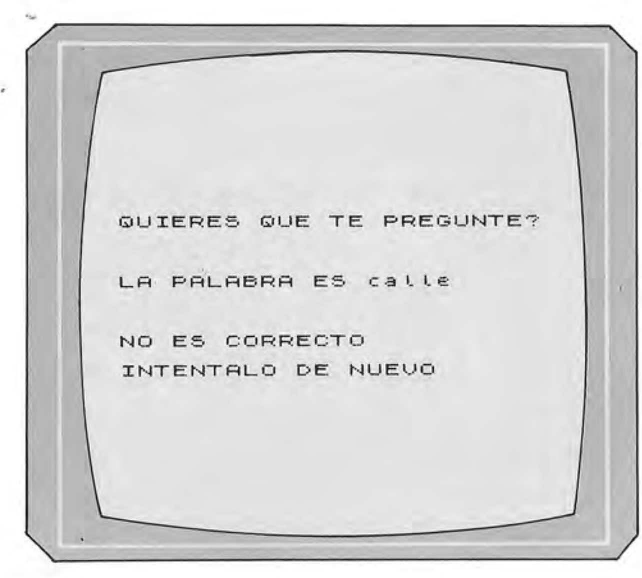
pueden ser más que suficientes, pero en la medida en que cada una vaya aprendiendo, o recordando, más palabras en francés, el programa se puede quedar pequeño. Para que el programa acepte más palabras sólo hay que ponerle más niveles. Para cambiar el número de niveles que pueda tener el programa sólo hay que cambiar las siguientes líneas:

```
215 CLS:INPUT "QUE NIVEL QUIERES
(1-n);L:IF L< OR L>N THEN GOTO 215
221 IF L=5 THEN LET D=320
222 IF L=6 THEN LET D=400
```

donde «n» es el número de nivel más alto permitido. Las líneas 221 y 222 aparecen a título de ejemplo. Habrá que poner una de estas líneas por cada nivel añadido, La asignación que se encuentra al final de dichas líneas le dice al ordenador cuál es el número máximo de palabras que puede preguntar.

Por supuesto, el cambio más importante ha de hacerse a partir de las líneas 350 y 450. A partir de la línea 350 tenemos que almacenar en líneas DATA las nuevas palabras en francés que queremos que tenga el programa. A partir de la línea 450 almacenaremos las mismas palabras, en el mismo orden, pero en español.






Pantalla del programa en plena ejecución.



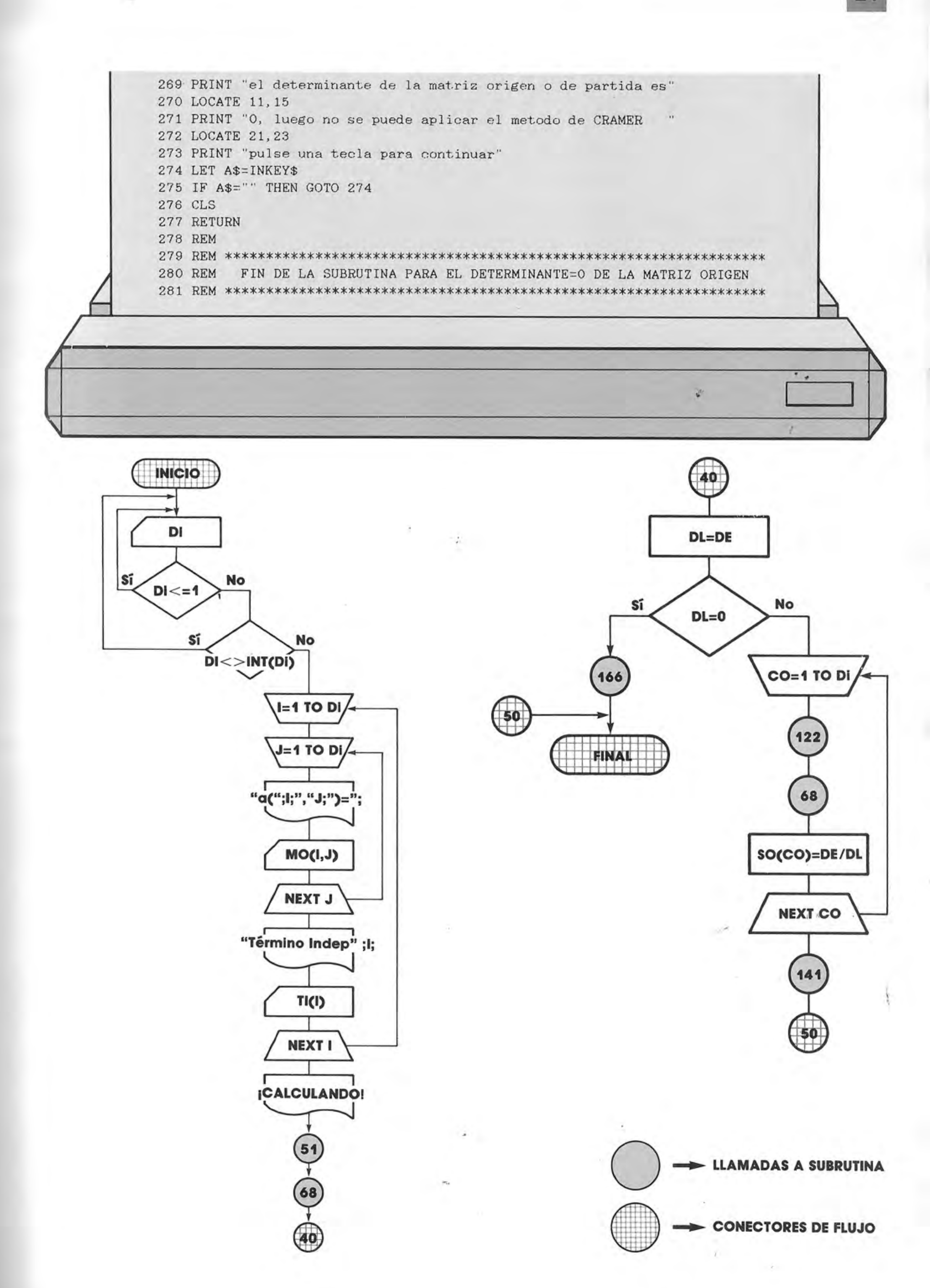
Programa: Resolución de ecuaciones

El programa que vamos a ver a continuación nos permitirá resolver ecuaciones que tengan hasta cincuenta incógnitas. El programa está preparado para decirnos si el determinante es cero, pero sólo sirve para ecuaciones que tengan una única solución. No sería difícil variar el programa para que además resolviese sistemas de ecuaciones con infinitas soluciones y para que resolviese sistemas de ecuaciones homogéneas. Para ello, se ha incluido, junto con el programa, el diagrama de flujo u organigrama.

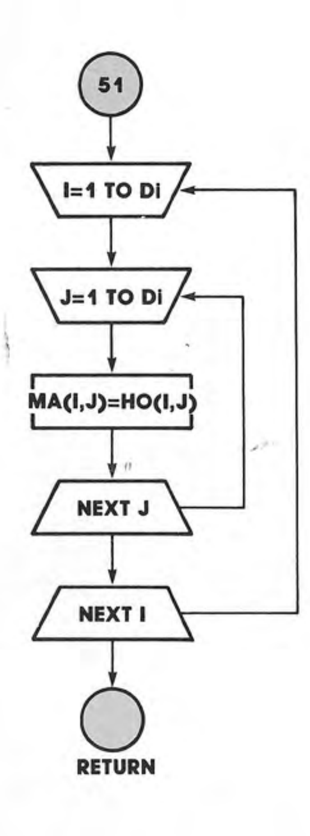
```
100 REM *****************************************************************
101 REM      RESOLUCION DE SISTEMAS DE ECUACIONES DE CUALQUIER ORDEN
102 REM      POR EL METODO DIRECTO DE  C R A M E R
103 REM *****************************************************************
104 REM
105 REM
106 REM
107 REM ****************************************************************
108 REM ***********      POR : JUAN MANUEL GUTIERREZ LEITON      *************
109 REM ****************************************************************
110 REM
111 REM
112 REM ****************************************************************
113 REM ************    (C) EDICIONES SIGLO CULTURAL (1987)    **************
114 REM ****************************************************************
115 REM
116 REM
117 DIM M(50,50):DIM A(50,50):DIM D(50):DIM T(50):DIM S(50)
118 CLS
119 INPUT "dimension del sistema de ecuaciones";DI
120 IF DI<=1 THEN GOTO 119
121 IF DI<>INT(DI) THEN GOTO 119
122 FOR I=1 TO DI
123    FOR J=1 TO DI
124       PRINT "a(";I;",";J;")=";
125       INPUT M(I,J)
126    NEXT J
127    PRINT "termino independiente(";I;")=";
128    INPUT T(I)
129 NEXT I
130 CLS
131 LOCATE 12,15
132 PRINT "-- CALCULANDO !!";
```

```
133 PRINT "        por favor, espere...."
134 GOSUB 147
135 GOSUB 164
136 LET DL=DE
137 IF DL=0 THEN GOTO 145
138 FOR C=1 TO DI
139    GOSUB 218
140    GOSUB 164
141    LET S(C)=DE/DL
142 NEXT C
143 GOSUB 237
144 GOTO 146
145 GOSUB 262
146 END
147 REM
148 REM ******************************************************************
149 REM   SUBRUTINA DE CREACION DE UNA MATRIZ AUXILIAR QUE ES UNA COPIA
150 REM   DE LA MATRIZ ORIGEN O MATRIZ DE PARTIDA DEL SISTEMA DE ECUACIO-
151 REM   NES.
152 REM ******************************************************************
153 REM
154 FOR I=1 TO DI
155    FOR J=1 TO DI
156    LET A(I,J)=M(I,J)
157    NEXT J
158 NEXT I
159 RETURN
160 REM
161 REM ******************************************************************
162 REM    FIN DE LA SUBRUTINA DE LA CREACION DE LA MATRIZ AUXILIAR
163 REM ******************************************************************
164 REM
165 REM ******************************************************************
166 REM    SUBRUTINA PARA EL CALCULO DE LOS DETERMINANTES POR EL METODO
167 REM    DE TRIANGULACION, ES DECIR, TODOS LOS ELEMENTOS SITUADOS ES-
168 REM    TRICTAMENTE BAJO LA DIAGONAL PRINCIPAL SON CONVERTIDOS A CERO
169 REM ******************************************************************
170 REM
171 LET T=1
172 LET I=T
173 LET J=T
174 LET K=T
175 LET L=T
176 IF M(I,J)=0 THEN GOTO 204
177 LET D(T)=M(K,J)
178 LET I=T
179 LET D=M(K,I)
180 LET M(I,J)=M(I,J)/D
181 IF J=DI THEN GOTO 184
182 LET J=J+1
183 GOTO 180
184 LET I=I+1
185 LET J=T
186 LET RE=M(I.J)
187 LET M(I,J)=M(I,J)-M(K,J)*RE
188 IF J=DI THEN GOTO 191
189 LET J=J+1
190 GOTO 187
191 IF I=DI THEN 193
192 GOTO 184
193 IF T=DI-1 THEN GOTO 196
194 LET T=T+1
195 GOTO 172
196 LET T=DI
197 LET D(T)=M(DI,DI)
198 LET T=1
199 LET DE=1
200 FOR I=1 TO DI
```

```
201    LET DE=DE*D(I)
202 NEXT I
203 GOTO 213
204 LET I=I+1
205 IF M(I,J)=0 THEN GOTO 210
206 LET M(K,L)=M(K,L)+M(I,L)
207 LET L=L+1
208 IF L>DI THEN GOTO 177
209 GOTO 206
210 IF I=DI THEN GOTO 212
211 GOTO 204
212 LET DE=0
213 RETURN
214 REM
215 REM ********************************************************************
216 REM          FIN DE LA SUBRUTINA DE CALCULO DE DETERMINANTES
217 REM ********************************************************************
218 REM
219 REM ********************************************************************
220 REM   SUBRUTINA PARA EL INTERCAMBIO DE LOS VALORES DE UNA COLUMNA
221 REM   DE LA MATRIZ ORIGEN POR LA COLUMNA DE TERMINOS INDEPENDIENTES
222 REM ********************************************************************
223 REM
224 FOR I=1 TO DI
225    FOR J=1 TO DI
226       M(I,J)=A(I,J)
227    NEXT J
228 NEXT I
229 FOR I=1 TO DI
230    M(I,C)=T(I)
231 NEXT I
232 RETURN
233 REM
234 REM *******************************************************************
235 REM          FIN DE LA SIBRUTINA DE INTERCAMBIO
236 REM *******************************************************************
237 REM
238 REM *******************************************************************
239 REM    SUBRUTINA PARA LA IMPRESION EN PANTALLA DE LOS RESULTADOS
240 REM *******************************************************************
241 REM
242 CLS
243 LOCATE 8,20
244 PRINT "soluciones del sistema de ecuaciones"
245 LOCATE 9,20
246 PRINT "------------------------------------"
247 FOR I=1 TO DI
248    LOCATE 13,30
249    PRINT "X";I;"=";S(I)
250    LOCATE 21,23
251    PRINT "pulse una tecla para continuar"
252    LET A$=INKEY$
253    IF A$="" THEN GOTO 252
254 NEXT I
255 CLS
256 RETURN
257 REM
258 REM *****************************************************************
259 REM    FIN DE LA SUBRUTINA DE IMPRESION DE RESULTADOS
260 REM *****************************************************************
261 REM
262 REM *****************************************************************
263 REM       SUBRUTINA PARA EL CASO DE QUE LA MATRIZ DE SALIDA O
264 REM       MATRIZ DE ORIGEN TENGA UN DETERMINANTE=0
265 REM *****************************************************************
266 REM
267 CLS
268 LOCATE 9,15
```

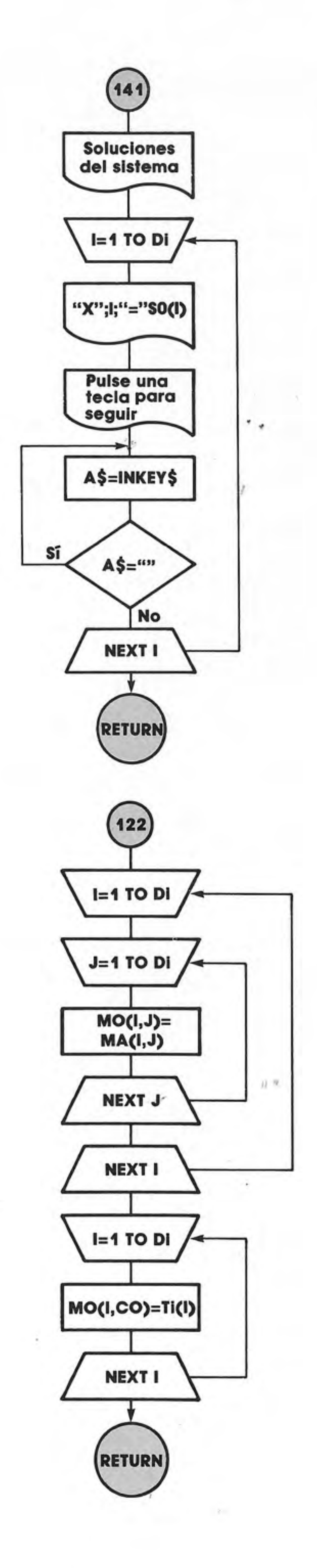


269 PRINT "el determinante de la matriz origen o de partida es"
270 LOCATE 11,15
271 PRINT "0, luego no se puede aplicar el metodo de CRAMER "
272 LOCATE 21,23
273 PRINT "pulse una tecla para continuar"
274 LET A$=INKEY$
275 IF A$="" THEN GOTO 274
276 CLS
277 RETURN
278 REM
279 REM **
280 REM FIN DE LA SUBRUTINA PARA EL DETERMINANTE=0 DE LA MATRIZ ORIGEN
281 REM **
INICIO
DI
Sí
DI<=1
No
Sí
No
DI<>INT(DI)
I=1 TO DI
J=1 TO DI
"a(";I;",";J;")=";
MO(I,J)
NEXT J
"Término Indep" ;I;
TI(I)
NEXT I
¡CALCULANDO!
51
68
40
40
DL=DE
Sí
DL=0
No
166
50
FINAL
CO=1 TO DI
122
68
SO(CO)=DE/DL
NEXT CO
141
50
LLAMADAS A SUBRUTINA
CONECTORES DE FLUJO

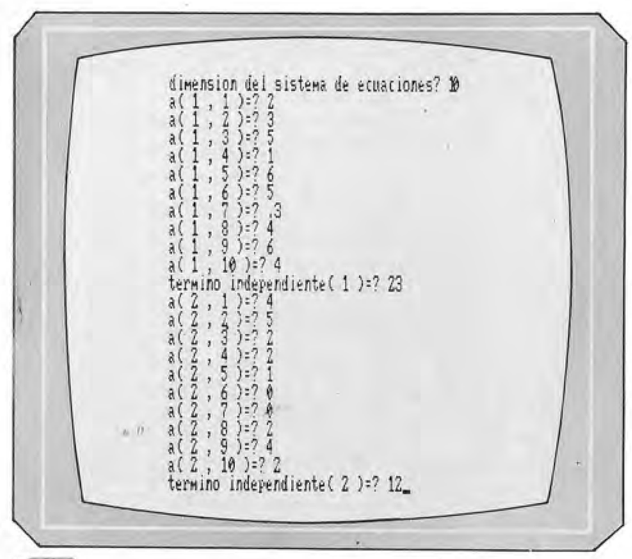


68
T=1
76
I=T
J=T
K=T
L=T
MO(I,J)=0
Sí
No
I=I+1
MO(I,J)=0
Sí
No
I=Di
NO
Sí
DE=0
117
RETURN
MO(K,L)=MO (K,L)+MO(I,L)
L=L+1
L>Di
No
Sí
DG(T)=MO(K,J)
I=T
D=MO(K,I)
MO(I,J)=MO (I,J)/D
J=Di
Sí
No
J=J+1
88
I=I+1
J=T
RE=MO(I,J)
MO(I,J)=MO(I,J) −MO(K,J)*RE
J=Di
Sí
No
J=J+1
I=Di
Sí
No
97
88

97
T=Di−1
No
Sí
T=T+1
76
T=Di
DG(T)=MO(Di,Di)
T=1
DE=1
T=1 TO Di
DE+DE*DG(I)
NEXT I
117



166
El sistema no tiene solución por CRAMER
Pulse una tecla para continuar
A$=INKEY$
Sí
A$=""
No
RETURN

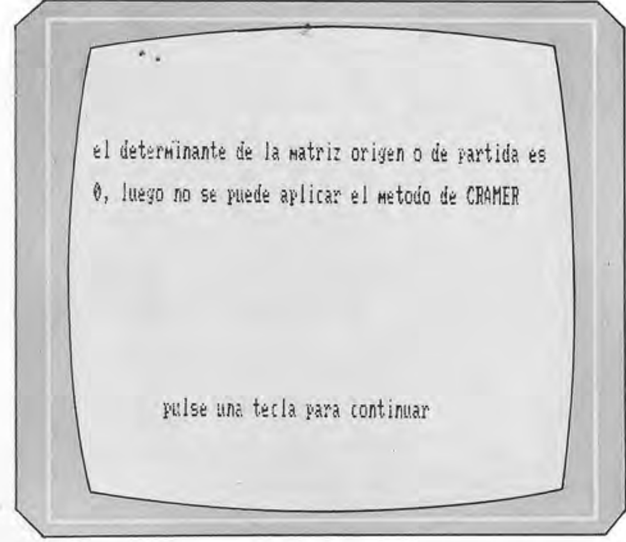



Ejemplo de ejecución.

soluciones del sistema de ecuaciones

X 1 = 3.857823

pulse una tecla para continuar




El programa avisa si el determinante de la matriz origen es cero.

Como siempre, para que el programa pueda funcionar perfectamente en ordenadores distintos del IBM, se recomienda realizar los siguientes cambios:

COMMODORE:

```
118 PRINT CHR$(147)
130 PRINT CHR$(147)
131 POKE 214,12:POKE 211,7
242 PRINT CHR$(147)
243 POKE 214,8:POKE 211,0
245 POKE 214,9:POKE 211,0
248 POKE 214,13:POKE 211,9
250 POKE 214,21:POKE 211,3
252 GET A$
255 PRINT CHR$(147)
268 POKE 214,9:POKE 211,0
270 POKE 214,11:POKE 211,0
172 POKE 214,21:POKE 211,3
274 GET A$
276 PRINT CHR$(147)
```

AMSTRAD:

```
131 LOCATE 15,12
243 LOCATE 20,8
245 LOCATE 20,9
248 LOCATE 30,13
250 LOCATE 23,21
268 LOCATE 15,9
270 LOCATE 15,11
272 LOCATE 23,21
```

Para que la presentación del programa sea perfecta, se recomienda al usuario ejecutar el programa estando en el modo número 2 de pantalla (MODE 2).

MXS:

```
131 LOCATE 15,12
242 LOCATE 20,8
245 LOCATE 20,9
248 LOCATE 30,13
250 LOCATE 23,21
268 LOCATE 15,9
270 LOCATE 15,11
272 LOCATE 23,21
```

SPECTRUM:

```
131 PRINT AT 12,0;
266 LET M(I,J)=A(I,J)
230 LET M(I,C)-T(I)
243 PRINT AT 8,0;
245 PRINT AT 9,0;
248 PRINT AT 13,0;
250 PRINT AT 20,0;
268 PRINT AT 9,0;
270 PRINT AT 11,0;
272 PRINT AT 20,0;
```

Notas sobre el programa: resolución de ecuaciones

Si necesitas que el programa te resuelva una ecuación de más de cincuenta incógnitas, lo único que tienes que cambiar es la línea 117 por:

117 DIM(n,n):DIM A(n,n):DIM D(n):DIM T(n):DIM S(n)

donde «n» es el número de incógnitas (y de ecuaciones) a resolver.

Para los usuarios del IBM, del AMSTRAD y del MSX2, se recomienda poner la pantalla en el modo de 80 columnas para tener una mejor presentación del programa.



Programa: Solitario para SPECTRUM

Este último programa del tomo es un juego de cartas que sólo sirve para el SPECTRUM, aunque aparecerá para el resto de los ordenadores en tomos sucesivos.

Con este programa podremos pasar nuestros ratos de ocio sin compañía de nadie, pues se trata de un solitario. Debido a la gran cantidad de juegos de este tipo que hay, se hace necesario explicar las instrucciones, no sólo las de uso del programa, sino también las del juego.

```
   1 REM ************************
   2 REM *      SOLITARIO       *
   3 REM ************************
   4 REM *POR J. GARCIA LUENGO *
   5 REM ************************
   6 REM *(c)Ed. Siglo Cultural*
   7 REM *(c)1987              *
   8 REM ************************
   9 REM
  10 RESTORE 5900: FOR A=5E4 TO 50011: READ B: POKE A,B: NEXT A: RESTORE 5910: F
OR A=50100 TO 50111: READ B: POKE A,B: NEXT A
  15 POKE 23658,8
  20 FOR A=1 TO 10: LET C=C+1: RESTORE 90: IF C=8 THEN  LET C=10
  30 FOR B=1 TO 4
  40 LET B$(A*4-4+B,1)=(STR$ C AND C<8 AND C>1)+("A" AND C=1)+("J" AND C=10)+("Q
" AND C=11)+("K" AND C=12)
  50 READ P$
  60 LET B$(A*4-4+B,2)=P$
  70 NEXT B
  80 NEXT A
  90 DATA "O","C","E","B"
 100 REM BARAJEO
 110 CLS : PRINT  FLASH 1;AT 11,11;"BARAJANDO"
 120 FOR A=1 TO 100
 130 LET ORDEN1=INT (RND*40)+1
 140 LET ORDEN2=INT (RND*40)+1
 150 IF ORDEN2=ORDEN1 THEN  GO TO 140
 160 LET P$=B$(ORDEN1)
 170 LET B$(ORDEN1)=B$(ORDEN2)
 180 LET B$(ORDEN2)=P$
 190 NEXT A
 200 REM CORTAR
 210 PRINT  FLASH 1;AT 11,11;"  CORTA  "
 220 FOR A=1 TO 40
 230 BEEP .01,30: PRINT AT 13,13;" ";A;" "
 240 IF INKEY$<>"" THEN  GO TO 260
 250 NEXT A: GO TO 200
 260 FOR B=1 TO 40
 270 LET C$(B)=B$(A)
 280 LET A=A+1: IF A=41 THEN  LET A=1
 290 NEXT B
 300 FOR A=1 TO 40
 310 LET B$(A)=C$(A)
 320 NEXT A
 330 REM CREA LA MESA
```

```
 335 CLS
 340 LET C=40
 350 LET D=6
 360 FOR B=1 TO 5
 370 LET A=1
 380 LET D=D-1
 390 LET A$(B,A)=B$(C)
 400 LET C=C-1
 410 LET A=A+1
 420 IF A<=D THEN  GO TO 390
 430 NEXT B
 440 LET D=7
 450 FOR B=0 TO 4
 460 LET A=1
 470 LET D=D-1
 480 PRINT AT A*2,B*3+2;A$(A,B+1)
 490 LET A=A+1
 500 IF A<>D THEN  GO TO 480
 510 NEXT B
 520 PRINT AT 1,19;"OROS";AT 3,19;R$(R)
 530 PRINT AT 1,25;"COPAS";AT 3,25;O$(O)
 540 PRINT AT 6,19;"ESPADAS";AT 8,19;E$(E)
 550 PRINT AT 9,24;"BASTOS";AT 11,24;S$(S)
 560 PRINT AT 13,19;"MAZO";: GO SUB 3960
 570 PRINT AT 16,24;"MONTON";AT 18,24;M$(M)
 580 PRINT AT 20,20; BRIGHT 1;"H=AYUDA"; BRIGHT 0
 590 LET CX=0
 600 GO SUB 800
 610 GO SUB 900
 615 IF C=0 AND M=1 THEN  IF R=11 AND O=11 AND E=11 AND S=11 THEN  GO TO 1600
 620 IF INKEY$="0" THEN  GO SUB 1000
 630 IF INKEY$="O" OR INKEY$="C" OR INKEY$="E" OR INKEY$="B" THEN  GO SUB 2000
 640 IF INKEY$="P" THEN  GO TO 3000
 650 IF INKEY$="R" THEN  IF C<>0 THEN  PRINT AT 13,19; FLASH 1;"MAZO"; FLASH 0:
GO SUB 4000
 660 IF INKEY$="N" THEN  IF M<>1 THEN  PRINT AT 16,24; FLASH 1;"MONTON"; FLASH 0
: GO SUB 5000
 670 IF INKEY$="Q" THEN  GO TO 970
 680 IF INKEY$="H" THEN  GO SUB 5600
 690 GO TO 600
 700 PRINT AT A*2,CXF*3+2;A$(A,CXF+1)
 710 RETURN
 750 PRINT AT FILAB*2,CXB*3+2;"  "
 760 RETURN
 800 IF INKEY$="8" THEN  PRINT AT 0,CX*3+2;"  ": BEEP .01,40: LET CX=CX+(1 AND C
X<>4)
 810 IF INKEY$="5" THEN  PRINT AT 0,CX*3+2;"  ": BEEP .01,40: LET CX=CX+(-1 AND
CX<>0)
 820 RETURN
 900 PRINT AT 0,CX*3+2; BRIGHT 1;"\/"; BRIGHT 0
 910 RETURN
 950 PRINT AT 0,CXB*3+2;"  "
 960 RETURN
 970 PRINT AT 21,0;"SI ESTAS SEGURO PULSA 'C'"
 975 LET K$=INKEY$
 980 IF K$="C" THEN  GO TO 10
 990 IF K$="" OR K$="Q" THEN  GO TO 975
 995 PRINT AT 21,0;"                                ": GO TO 600
1000 LET CXB=CX
1010 GO SUB 900
1020 PRINT AT 0,CXB*3+2;"\/"
1030 GO SUB 800
1040 IF INKEY$<>"1" THEN  GO TO 1010
1050 LET CXF=CX
1060 LET X=CXB
1070 GO SUB 1500
1075 IF A=1 THEN  GO SUB 950: RETURN
1080 LET FILAB=A-1
```

```
1090 LET X=CXF
1095 GO SUB 1500: IF A=1 THEN  GO SUB 950: GO TO 1210
1100 IF A=11 THEN  IF A$(A,X+1)<>"  " THEN  RETURN
1105 IF A$(A,CXF+1)="  " THEN  IF A=1 THEN  GO TO 1210
1110 GO SUB 1200
1120 GO SUB 950
1130 RETURN
1200 IF A=1 THEN  GO TO 1210
1205 GO SUB 1300: IF A$(A-1,CXF+1,2)=A$(FILAB,CXB+1,2) OR FINAL<=BASE OR FINAL<>
BASE+1 THEN  RETURN
1210 LET A$(A,CXF+1)=A$(FILAB,CXB+1)
1220 LET A$(FILAB,CXB+1)="  "
1230 GO SUB 750
1235 GO SUB 700
1290 RETURN
1300 LET P$=A$(FILAB,CXB+1,1)
1310 GO SUB 1400
1315 LET BASE=FINAL
1320 LET P$=A$(A-1,CXF+1,1)
1330 GO SUB 1400
1340 RETURN
1400 LET FINAL=(1 AND P$="A")+(2 AND P$="2")+(3 AND P$="3")+(4 AND P$="4")+(5 AN
D P$="5")+(6 AND P$="6")+(7 AND P$="7")+(8 AND P$="J")+(9 AND P$="Q")+(10 AND P$
="K")
1410 RETURN
1500 FOR A=1 TO 10
1510 IF A$(A,X+1)="  " THEN  RETURN
1520 NEXT A
1530 RETURN
1600 PRINT'AT 21,0; FLASH 1;"L O  C O N S E G U I S T E . . ."; FLASH 0
1610 FOR A=-40 TO 40: BEEP .01,A
1620 IF INKEY$<>"" THEN  GO TO 1630
1625 NEXT A: IF INKEY$="" THEN  GO TO 1610
1630 PAUSE 0
1640 CLS
1650 GO TO 100
2000 IF INKEY$="O" THEN  GO TO 2200
2010 IF INKEY$="C" THEN  GO TO 2400
2020 IF INKEY$="E" THEN  GO TO 2600
2030 IF INKEY$="B" THEN  GO TO 2800
2040 GO TO 2000
2200 LET X=CX
2210 GO SUB 1500
2220 IF A<>1 THEN  LET A=A-1
2230 IF R$(R)="  " THEN  IF A$(A,CX+1,1)="A" THEN  GO SUB 2310
2240 IF R$(R)="  " THEN  IF A$(A,CX+1,1)<>"A" THEN  RETURN
2250 LET P$=R$(R,1)
2260 GO SUB 1400
2270 LET BASE=FINAL
2280 LET P$=A$(A,CX+1,1)
2290 GO SUB 1400
2300 IF BASE<>FINAL-1 THEN  RETURN
2310 LET R=R+1: GO SUB 2350
2320 RETURN
2350 LET R$(R)=A$(A,CX+1): IF R$(R,2)<>"O" THEN  LET R$(R)="  ": LET R=R-1: RETU
RN
2360 GO SUB 2370: LET A$(A,CX+1)="  ": RETURN
2370 PRINT AT 3,19;R$(R)
2380 LET FILAB=A: LET CXB=CX: GO SUB 750
2390 RETURN
2400 LET X=CX
2410 GO SUB 1500
2420 IF A<>1 THEN  LET A=A-1
2430 IF O$(O)="  " THEN  IF A$(A,CX+1,1)="A" THEN  GO TO 2510
2440 IF O$(O)="  " THEN  IF A$(A,CX+1,1)<>"A" THEN  RETURN
2450 LET P$=O$(O,1)
2460 GO SUB 1400
2470 LET BASE=FINAL
```

```
2480 LET P$=A$(A,CX+1,1)
2490 GO SUB 1400
2500 IF BASE<>FINAL-1 THEN  RETURN
2510 LET O=O+1: GO SUB 2550
2520 RETURN
2550 LET O$(O)=A$(A,CX+1): IF O$(O,2)<>"C" THEN  LET O=O-1: RETURN
2560 GO SUB 2570: LET A$(A,CX+1)="  ": RETURN
2570 PRINT AT 3,25;O$(O)
2580 LET FILAB=A: LET CXB=CX: GO SUB 750
2590 RETURN
2600 LET X=CX
2610 GO SUB 1500
2620 IF A<>1 THEN  LET A=A-1
2630 IF E$(E)="  " THEN  IF A$(A,CX+1,1)="A" THEN  GO TO 2710
2640 IF E$(E)="  " THEN  IF A$(A,CX+1,1)<>"A" THEN  RETURN
2650 LET P$=E$(E,1)
2660 GO SUB 1400
2670 LET BASE=FINAL
2680 LET P$=A$(A,CX+1,1)
2690 GO SUB 1400
2700 IF BASE<>FINAL-1 THEN  RETURN
2710 LET E=E+1: GO SUB 2750
2720 RETURN
2750 LET E$(E)=A$(A,CX+1): IF E$(E,2)<>"E" THEN  LET E=E-1: RETURN
2760 GO SUB 2770: LET A$(A,CX+1)="  ": RETURN
2770 PRINT AT 8,19;E$(E)
2780 LET FILAB=A: LET CXB=CX: GO SUB 750
2790 RETURN
2800 LET X=CX
2810 GO SUB 1500
2820 IF A<>1 THEN  LET A=A-1
2830 IF S$(S)="  " THEN  IF A$(A,CX+1,1)="A" THEN  GO TO 2910
2840 IF S$(S)="  " THEN  IF A$(A,CX+1,1)<>"A" THEN  RETURN
2850 LET P$=S$(S,1)
2860 GO SUB 1400
2870 LET BASE=FINAL
2880 LET P$=A$(A,CX+1,1)
2890 GO SUB 1400
2900 IF BASE<>FINAL-1 THEN  RETURN
2910 LET S=S+1: GO SUB 2950
2920 RETURN
2950 LET S$(S)=A$(A,CX+1): IF S$(S,2)<>"B" THEN  LET S=S-1: RETURN
2960 GO SUB 2970: LET A$(A,CX+1)="  ": RETURN
2970 PRINT AT 11,24;S$(S)
2980 LET FILAB=A: LET CXB=CX: GO SUB 750
2990 RETURN
3000 REM PALO A MESA
3010 IF INKEY$="O" THEN  GO TO 3100
3020 IF INKEY$="C" THEN  GO TO 3250
3030 IF INKEY$="E" THEN  GO TO 3400
3040 IF INKEY$="B" THEN  GO TO 3550
3050 GO TO 3000
3100 IF R$(R)="  " THEN  GO TO 600
3110 LET X=CX
3120 GO SUB 1500: IF A=11 THEN  IF A$(A-1,CX+1)<>"  " THEN  GO TO 600
3125 IF A=1 THEN  GO TO 3150
3130 LET P$=R$(R)
3140 GO SUB 3900
3150 LET A$(A,CX+1)=R$(R)
3160 LET R$(R)="  "
3170 IF R<>1 THEN  LET R=R-1
3180 LET CXF=CX
3190 GO SUB 700
3200 PRINT AT 3,19;R$(R)
3210 GO TO 600
3250 IF O$(O)="  " THEN  GO TO 600
3260 LET X=CX
3270 GO SUB 1500: IF A=11 THEN  IF A$(A-1,CX+1)<>"  " THEN  GO TO 600
```

```
3275 IF A=1 THEN  GO TO 3300
3280 LET P$=O$(O)
3290 GO SUB 3900
3300 LET A$(A,CX+1)=O$(O)
3310 LET O$(O)="  "
3320 IF O<>1 THEN  LET O=O-1
3330 LET CXF=CX
3340 GO SUB 700
3350 PRINT AT 3,25;O$(O)
3360 GO TO 600
3400 IF E$(E)="  " THEN  GO TO 600
3410 LET X=CX
3420 GO SUB 1500: IF A=11 THEN  IF A$(A-1,CX+1)<>"  " THEN  GO TO 600
3425 IF A=1 THEN  GO TO 3450
3430 LET P$=E$(E)
3440 GO SUB 3900
3450 LET A$(A,CX+1)=E$(E)
3460 LET E$(E)="  "
3470 IF E<>1 THEN  LET E=E-1
3480 LET CXF=CX
3490 GO SUB 700
3500 PRINT AT 8,19;E$(E)
3510 GO TO 600
3550 IF S$(S)="  " THEN  GO TO 600
3560 LET X=CX
3570 GO SUB 1500: IF A=11 THEN  IF A$(A-1,CX+1)<>"  " THEN  GO TO 600
3575 IF A=1 THEN  GO TO 3600
3580 LET P$=S$(S)
3590 GO SUB 3900
3600 LET A$(A,CX+1)=S$(S)
3610 LET S$(S)="  "
3620 IF S<>1 THEN  LET S=S-1
3630 LET CXF=CX
3640 GO SUB 700
3650 PRINT AT 11,24;S$(S)
3660 GO TO 600
3700 PRINT AT 1,19;"OROS";AT 1,25;"COPAS";AT 6,19;"ESPADAS";AT 9,24;"BASTOS"
3710 RETURN
3900 IF A$(A-1,CX+1,2)=P$(2) THEN  GO TO 600
3905 LET P$=P$(1)
3910 GO SUB 1400
3915 LET BASE=FINAL
3920 LET P$=A$(A-1,CX+1,1)
3930 GO SUB 1400
3940 IF FINAL<>BASE+1 THEN  GO TO 600
3950 RETURN
3960 PRINT AT 15,19;"##"; OVER 1;CHR$ 8;CHR$ 8;"__"
3970 PRINT AT 13,19;"MAZO= ";C;" "
3980 RETURN
4000 PRINT AT 15,19;B$(C)
4010 IF INKEY$="M" THEN  GO TO 4100
4020 IF INKEY$="T" THEN  GO SUB 4200: IF A$(A,CX+1)=B$(C+1) THEN  RETURN
4030 IF INKEY$="O" OR INKEY$="C" OR INKEY$="E" OR INKEY$="B" THEN  GO TO 4300
4040 GO TO 4000
4100 LET M$(M)=B$(C)
4110 PRINT AT 18,24;M$(M)
4120 LET M=M+1
4130 LET C=C-1
4140 GO SUB 3960
4150 RETURN
4200 GO SUB 900: GO SUB 800
4210 IF INKEY$<>"1" THEN  GO TO 4200
4220 LET X=CX
4230 GO SUB 1500: LET A$(A,CX+1)=B$(C): IF A=11 THEN  IF A$(A,CX+1)<>"  " THEN
RETURN
4235 IF A=1 THEN  GO TO 4250
4240 GO SUB 4500: IF BASE>=FINAL OR BASE<>FINAL-1 OR A$(A,CX+1,2)=A$(A-1,CX+1,2)
 THEN  LET A$(A,CX+1)="  ": RETURN
4250 LET C=C-1
```

```
4260 LET CXF=CX
4270 GO SUB 700
4280 GO SUB 3960
4290 RETURN
4300 IF INKEY$="O" OR INKEY$="C" OR INKEY$="E" OR INKEY$="B" THEN  GO TO 4320
4310 GO TO 4300
4320 LET X=CX
4330 GO SUB 1500: IF A=11 THEN  IF A$(A,CX+1)<>"  " THEN  RETURN
4340 LET A$(A,CX+1)=B$(C)
4350 GO SUB 2000
4360 IF R$(R)=B$(C) OR O$(O)=B$(C) OR E$(E)=B$(C) OR S$(S)=B$(C) THEN  LET C=C-1
: GO SUB 3960: RETURN
4370 LET A$(A,CX+1)="  "
4380 GO TO 4000
4500 LET A$(A,CX+1)=B$(C)
4510 LET FILAB=A
4520 LET CXF=CX
4530 LET CXB=CX
4540 GO SUB 1300
4550 RETURN
5000 IF INKEY$="T" THEN  GO TO 5100
5010 IF INKEY$="O" OR INKEY$="C" OR INKEY$="E" OR INKEY$="B" THEN  GO TO 5200
5020 GO TO 5000
5100 GO SUB 900
5110 GO SUB 800
5120 IF INKEY$<>"1" THEN  GO TO 5100
5130 LET X=CX
5140 GO SUB 1500: IF A=11 AND A$(A,CX+1)<>"  " THEN  PRINT AT 16,24;"MONTON": RE
TURN
5150 IF A=1 THEN  LET A$(A,CX+1)=M$(M-1): GO TO 5160
5155 GO SUB 5500: IF BASE>=FINAL OR BASE<>FINAL-1 OR A$(A,CX+1,2)=A$(A-1,CX+1,2)
 THEN  LET A$(A,CX+1)="  ": GO TO 5320
5160 LET M=M-1
5170 LET CXF=CX
5180 GO SUB 700
5190 GO TO 5300
5200 IF INKEY$="O" OR INKEY$="C" OR INKEY$="E" OR INKEY$="B" THEN  GO TO 5220
5210 GO TO 5200
5220 LET X=CX
5230 GO SUB 1500
5240 LET A$(A,CX+1)=M$(M-1)
5250 GO SUB 2000
5260 IF R$(R)=M$(M-1) OR O$(O)=M$(M-1) OR E$(E)=M$(M-1) OR S$(S)=M$(M-1) THEN  L
ET M=M-1: LET A$(A,CX+1)="  ": GO TO 5300
5270 GO TO 5320: LET A$(A,CX+1)="  "
5280 RETURN
5300 LET M$(M)="  ": IF M=1 THEN  PRINT AT 18,24;"  ": GO TO 5320
5310 PRINT AT 18,24;M$(M-1)
5320 PRINT AT 16,24;"MONTON"
5330 RETURN
5500 LET A$(A,CX+1)=M$(M-1)
5510 GO TO 4510
5600 RANDOMIZE USR 5E4: CLS
5610 PRINT AT 0,0;"5..........CURSOR A LA IZQUIERDA"
5620 PRINT "6..........CURSOR A LA DERECHA"
5630 PRINT "0..........COGER UNA CARTA"
5640 PRINT "1..........SOLTAR UNA CARTA"
5650 PRINT "R..........ROBAR CARTA DEL MAZO"
5660 PRINT " M.........PASAR CARTA AL MONTON"
5670 PRINT " T.........PASAR CARTA A LA MESA"
5680 PRINT "  1........SOLTAR CARTA EN MESA"
5690 PRINT " O,C,E,B...PASAR CARTA AL PALO"
5700 PRINT "N..........PASA CARTA DEL MONTON"
5710 PRINT " T.........A LA MESA"
5720 PRINT "  1........SOLTAR CARTA EN MESA"
5730 PRINT " O,C,E,B...AL PALO"
5740 PRINT "P.........PASA CARTA DEL PALO"
```

```
5750 PRINT " T.........A LA MESA"
5760 PRINT "  1........SOLTAR CARTA EN MESA"
5770 PRINT "Q..........RENDIRSE"
5780 PRINT AT 21,0; FLASH 1;"PULSA UNA TECLA PARA SEGUIR."; FLASH 0
5790 PAUSE 0
5800 RANDOMIZE USR 50100
5810 RETURN
5900 DATA 33,0,64,17,214,216,1,0,27,237,176,201
5910 DATA 33,214,216,17,0,64,1,0,27,237,176,201
```

```
  10>CLEAR :DIM A$(11,5,2):DIM B$(40,2):DIM C$(40,2):DIM E$(11,2):DIM O$(11,2):
DIM R$(11,2):DIM S$(11,2):DIM M$(30,2):LET R=1:LET O=1:LET E=1:LET S=1:LET M=1:L
ET C=0
  20 FOR A=1 TO 10: LET C=C+1: RESTORE 90: IF C=8 THEN  LET C=10
```

Empezaremos con las de uso del programa.

Una vez que hayas metido el programa en tu SPECTRUM y hayas hecho RUN, la pantalla del ordenador se volverá blanca y al rato aparecerá un mensaje en FLASH que te indicará que el programa está barajando las cartas. Tras unos segundos aparecerá un mensaje que te dice que cortes y unos números que se suceden a gran velocidad. En este momento tienes que pulsar una tecla para que el programa corte.

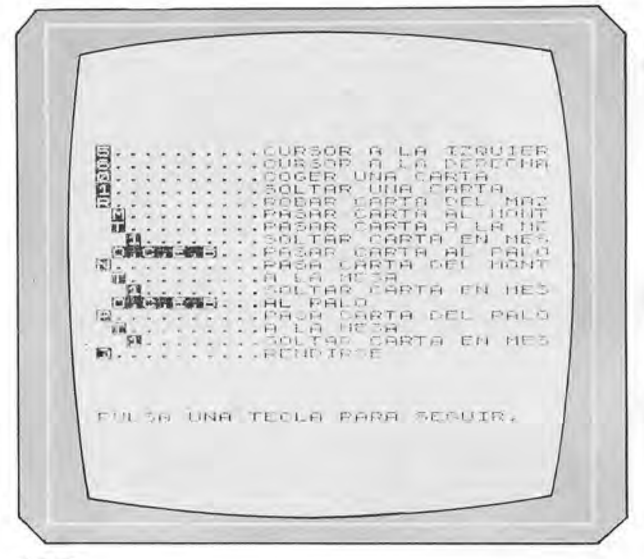
Una vez llegados a este punto aparecerá en pantalla el tablero con las cartas. A partir de este momento puedes empezar a jugar. Si no te acuerdas de todas las teclas que puedes utilizar en el programa, pulsa la tecla «H» y te aparecerá un resumen de ellas por pantalla.

El objetivo del juego es formar cada palo desde el AS hasta el REY.

Al principio del juego, y después de cortar, te aparecerán las cartas en la pantalla de arriba a abajo y de izquierda a derecha. Las cartas sobrantes formarán el mazo.

La forma de jugar es la siguiente:

1. Se reordenan las cartas que inicialmente están en la mesa. Con las teclas 5 y 8 se mueve el cursor superior (5 = izquierda, 8 = derecha). El cursor apunta a una columna y, en esta columna, a la carta situada más bajo. Para coger carta tienes que pulsar el 0, después de mover el cursor hasta la columna deseada y pulsar el 1 para soltar la carta. Si quieres dejar la carta donde estaba tienes que pulsar el 1 con el cursor en la misma columna. La carta se puede mover de una a otra columna si la carta de la columna inicial es de distinto palo que la carta de la columna final. Además, la carta inicial debe ser un punto más baja que la final, es decir, que si la carta inicial es un 7 y la final una SOTA, podrás poner el 7 sobre la sota.



 *El programa ejecutante.*

2. Robar del mazo pulsando la R. La carta superior del mazo se levantará pudiendo hacer con ella lo siguiente:

a) Pasarla a la mesa pulsando I y con-

trolando con 5 y 8 el cursor. Pulsar 1 para dejar la carta.

b) Pasarla al palo correspondiente pulsando O, C, E o B, según sea el palo de la carta. Si no hay ninguna carta en el palo, la primera tiene que ser un AS.

c) Pasarla al montón pulsando H. El montón está formado por las cartas que, o bien no podemos poner en la mesa, o bien no nos conviene ponerla en la mesa.

3. Pasar carta de la mesa al palo correspondiente situando el cursor en la columna donde está la carta y pulsar O, C, E o B según sea el palo de la carta. Las cartas hay que pasarlas al mazo por orden, no pudiendo poner, por ejemplo, el 5 de copas si no está el 4 de copas.

4. Pasar carta del montón al:

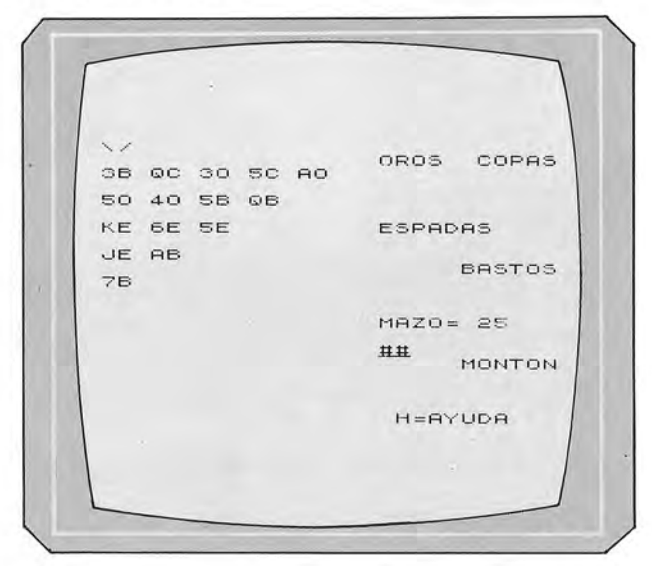
a) Palo pulsando N y después O, C, E o B.

b) A la mesa, pulsando N primero y después de I situar el cursor en la columna deseada y pulsar 1 para soltarla.

5. Pasar del palo a la mesa pulsado primero P y después O, C, E o B para situar la carta, si es posible, en la posición del cursor en la mesa.

La tecla Q nos servirá para rendirse y empezar de nuevo.

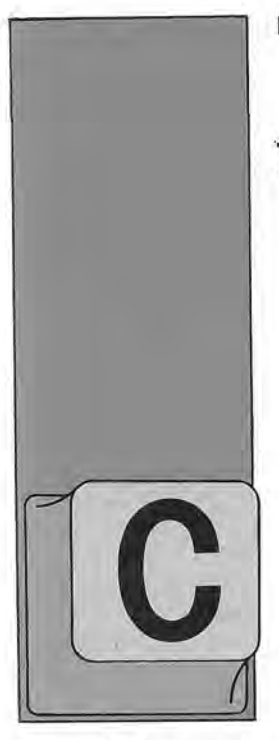
Cuando la carta superior de cada palo es el rey, entonces habrás terminado con éxito la partida.




Instrucciones del programa

TECNICAS DE ANALISIS

TABLAS DE DECISION (y III)



C OMO conclusión de los conceptos expuestos sobre las tablas de decisión, presentamos las normas que suelen seguirse en el análisis de las tablas, según el tipo de reglas que se han utilizado.

Hay que controlar dos aspectos: la existencia de redundancias (para eliminarlas adecuadamente) y el número total de reglas (para asegurarnos de que la tabla es completa).

En todos los casos se supone que se cumplen las siguientes condiciones:

a) La tabla de decisión es limitada (es decir, todas y cada una de las condiciones presentes en la tabla pueden evaluarse en forma afirmativa o negativa). Si la tabla no era originalmente limitada, ha de convertirse en limitada para hacer las evaluaciones que indicamos a continuación.

b) Cada regla de la tabla de decisión conduce a un tratamiento.

c) Las condiciones presentes son independientes entre sí (es decir, el que se cumpla una condición en una situación determinada no significa que se cumpla ninguna otra).

Vamos a denotar por «n» el número de condiciones que aparecen en la tabla de decisión, independientemente del número de tratamientos que aparezcan.



Análisis de tablas de decisión con reglas AND

Se examinarán dos aspectos:

a) Eliminación de redundancias:

Para ello, se comparan la reglas de 2 en 2. Las funciones de decisión han de ser distintas al menos para una evaluación de alguna de las condiciones (normalmente serán distintas en varios de los casos posibles). Se produce la redundancia cuando hay dos reglas que no contienen esa diferencia de evaluación y, por tanto, habrá que eliminar una de ellas. Puede suceder que, en el análisis anterior, se detecte que dos condiciones que se tenía por distintas son, en realidad iguales o dependientes; si, como resultado de esta evaluación, se modifica el conjunto de condiciones definido, es imprescindible rehacer todo el examen de redundancias desde el principio, para asegurar la consistencia del análisis realizado. En ocasiones, incluso dos reglas son redundantes pero los tratamientos a que dan lugar son distintos (zona inferior de la tabla): en ese caso, se dice que hay contradicción entre reglas y es imprescindible la eliminación de una de ellas (una vez examinada cuidadosamente la funcionalidad de ambas).

b) Control de completud:

Para que la tabla de decisión sea completa, el número de combinaciones entre las condiciones que se puede dar (2^n) ha de ser igual al número de reglas que aparecen en la tabla, teniendo en cuenta las indiferencias existentes (casos en que la condición ni ha de cumplirse ni dejar de cumplirse: se esciben como: – o como *).

Para tener en cuenta las indiferencias, las reglas que contienen una sola (un solo – en la tabla) se computan como 2. Si una regla tiene dos indiferencias (dos

– en su columna correspondiente), se cuentan cuatro posibles casos para esa regla: puede pensarse que cada una de las indiferencias puede ser S o N y, por tanto, hay $2^2 = 4$ posibles combinaciones (SS, SN, NS, NN). Si una regla tiene tres indiferencias, se cuenta como 8 (2^3) pues pueden darse ocho combinaciones para esa regla, según los valores que tomen esas indiferencias (SSS, SSN, SNS, SNN, NSS, etc.). Así pues, si denotamos por $N(=2^n)$ el número de condiciones presentes en las reglas AND, para que la tabla sea completa ha de cumplirse que

$$N = \sum_{i=0}^{m} 2^i \cdot N_i$$

donde m es el número máximo de indiferencias que aparece en cualquiera de las reglas de la tabla de decisión. Obsérvese que los primeros sumandos de la suma anterior son: $2^0 \cdot N_0 = 1 \cdot N_0 = N_0$ (número de reglas simples, es decir, sin indiferencias); $2^1 \cdot N_1 = 2 \cdot N_1$ (donde N_1 es el número de reglas que tiene una indiferencia); $2^2 \cdot N_2 = 4 \cdot N_2$ (N_2 es el número de reglas presentes con dos indiferencias), etcétera.

Si en la tabla de decisión hay una regla ELSE, y esta regla agrupa N_E reglas, la completud de la tabla viene dada por

$$N = \sum_{i=0}^{m} 2^i \cdot N_i + N_E$$



Análisis de tablas de decisión con reglas AND y OR

a) Eliminación de redundancias.

Hay que comparar dos a dos las reglas AND con las OR y las OR entre sí.

Cuando se compara una regla AND con otra OR, se comprueba que no hay redundancia si las dos funciones presentan evaluaciones contrarias para cada condición, excepto en el caso en que en alguna de las funciones presente una marca de indiferencia.

En el caso de la comparación entre reglas OR, no habrá redundancia o dependencia cuando las funciones de ambas presentan una anotación * en cada una de sus condiciones, excepto en una, que debe ser la misma para las dos y con evaluaciones distintas.

b) Control de completud.

Las reglas de tipo AND se cuentan como se ha descrito anteriormente.

En las de tipo OR, hay que multiplicar el número de reglas con «j» ocurrencias por un coeficiente (como en el caso AND) que en este caso es $2^n - 2^j$: es decir, de las 2^n posibles combinaciones que se pueden establecer en cada regla, hay que descontar 2^j que no se obtienen por la aparición de «j» indiferencias. Por tanto, la suma de todas las posibilidades con reglas OR viene dada por la expresión

$$\sum_{j=0}^{p} (2^n - 2^j) \cdot N_j$$

Obsérvese que $2^n = N$ es el número de condiciones posibles y los primeros sumandos de la suma anterior son $(2^n - 1) \cdot N_0$ (donde N_0 es el número de reglas OR con cero indiferencias), $(2^n - 2) \cdot N_1$ (para las reglas con una indiferencia), etc.

La suma total de las reglas para el cálculo de la completud será (para esta situación en que existen reglas AND y reglas OR),

$$N = \sum_{i=0}^{m} 2^i \cdot N_i + N_E' + \sum_{j=0}^{p} 2^j \cdot N_j$$



Análisis de tablas de decisión con condiciones dependientes

En este caso, la dependencia entre condiciones tiene lugar únicamente entre reglas AND, ya que las reglas OR no pueden contradecirse entre sí.

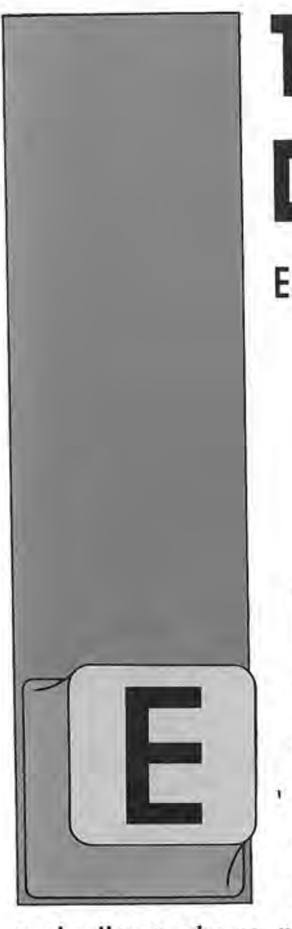
El cálculo de reglas total se hace como en los casos anteriores y las condiciones de redundancia no varían.

En este caso, no existen 2^n combinaciones (el número N antes considerado) ya que entre dos o más condiciones no se formarían todas las combinaciones posibles.

Si consideramos que hay G_2 grupos de dos condiciones dependientes entre sí y G_3 grupos de tres condiciones, etc., el factos a introducir será $K \cdot G_K$, de modo que el número total de reglas será $\prod_{K=0}^{l} K \cdot G_K$ donde «l» es el número de condiciones dependientes entre sí que existan en el grupo de condiciones que tenga más.

TECNICAS DE PROGRAMACION

ESTRUCTURA DE DATOS





Matrices o tablas

EN el capítulo anterior hemos visto cómo pueden proporcionársele a los programas datos aislados (escalares) o colecciones de datos (vectores o series). Vamos a dedicar ahora nuestra atención a estructuras algo más complejas, que reciben el nombre de matrices o «tablas».

Una tabla o matriz es un conjunto de datos dispuestos a lo largo de dos dimensiones, en forma de rectángulo. Veamos un ejemplo:

1	2	3	4
5	6	7	8
9	10	11	12

Siempre es posible considerar una tabla como una colección de vectores o series de datos. En el ejemplo anterior, podemos ver fácilmente que la matriz se compone de las tres series siguientes: (1, 2, 3, 4), (5, 6, 7, 8) y (9, 10, 11, 12). Estas tres series se llaman las «filas» de la matriz. Al leer sucesivamente las filas de una matriz, enunciamos todos los elementos de ésta en un orden determinado, de izquierda a derecha y de arriba a abajo: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12. Se dice, en este caso, que la matriz ha sido leída «por filas».

Por otra parte, y puesto que la tabla es rectangular, también podemos considerarla como una colección de las cuatro series o vectores siguientes: (1, 5, 9), (2, 6, 10), (3, 7, 11), (4, 8, 12). Cada una de estas series es una «columna» de la matriz. Además, al leer sucesivamente las columnas de la matriz, enunciamos sus elementos de arriba a abajo y de derecha a izquierda (al estilo japonés): 1, 5, 9, 2, 6, 10, 3, 7, 11, 4, 8, 12. Decimos entonces que la matriz ha sido leída «por columnas».

Por lo que acabamos de ver, está claro que la matriz del ejemplo tiene tres filas y cuatro columnas. El número total de datos que contiene es precisamente igual al producto del número de sus filas por el número de sus columnas. Es decir: $3 \times 4 = 12$. Es fácil comprobar que, en efecto, la tabla anterior tiene doce datos.

Veamos algunos ejemplos más. La tabla

1	2
3	4
5	6
7	8
9	0

tiene cinco filas y dos columnas. Vemos, pues, que es posible que una matriz tenga menos columnas que filas, lo que le dará una forma alargada verticalmente.

La tabla

1	2	3
4	5	6
7	8	9

tiene tres filas y tres columnas. Por razones obvias, cuando una matriz tiene el

mismo número de filas que de columnas, se dice que es «cuadrada». En caso contrario, se dice que es «rectangular».

La tabla

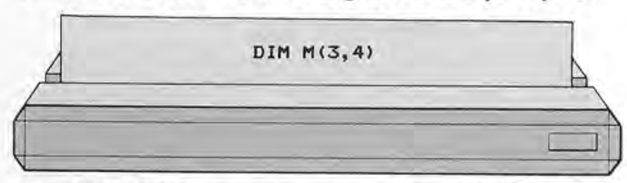
1
2
3
4

es un poco especial. Tiene cuatro filas: (1), (2), (3) y (4), pero una sola columna: (1, 2, 3, 4). Finalmente, la tabla

1	2	3	4

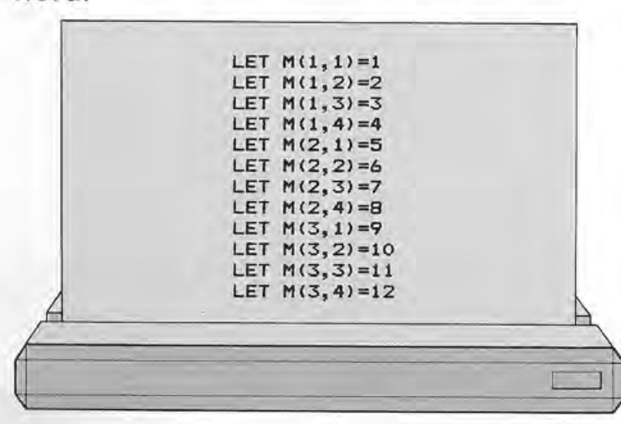
tiene una sola fila: (1, 2, 3, 4) y cuatro columnas: (1), (2), (3) y (4). Es fácil darse cuenta de que esta última tabla puede considerarse también como una serie o vector de cuatro elementos.

Las tablas o matrices se usan frecuentemente en los programas, pues no es raro que los datos que hemos de utilizar se distribuyan de forma natural en una disposición rectangular. En el lenguaje BASIC, una tabla debe declararse, al igual que un vector, mediante una instrucción DIM, pero en lugar de indicar el número de sus elementos, daremos el número de sus filas y el número de sus columnas, como en el siguiente ejemplo:

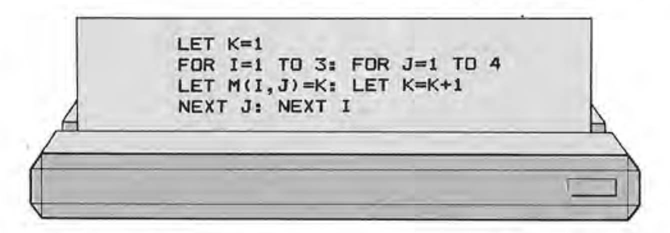


donde declaramos que la variable M es una tabla de tres filas y cuatro columnas, como la de nuestro primer ejemplo.

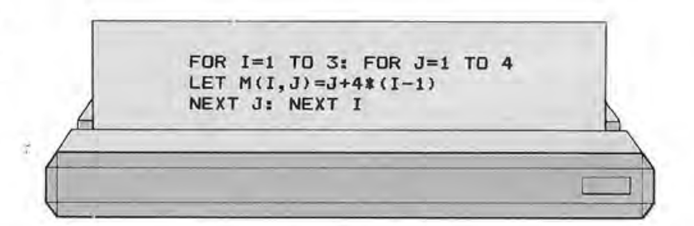
¡Bien! Ya tenemos definida la matriz. Pero ahora falta llenar sus casillas con los doce valores correspondientes. En BASIC esto se puede hacer de la siguiente manera:



En el caso concreto de nuestro ejemplo, podemos llenar la matriz de otra manera, más rápida, haciendo uso del hecho de que los doce valores son consecutivos y utilizando dos bucles: uno sobre las filas y otro sobre las columnas. El programa correspondiente quedará así:

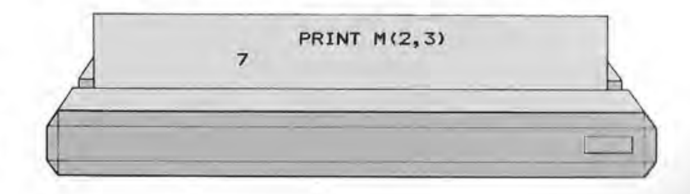


Explicaremos este programa cuando hablemos de los bucles. Por el momento, basta con ver que, efectivamente, el programa 3 es totalmente equivalente al 2, pues realiza la misma función. Eso puede comprobarse ejecutándolo en cualquier ordenador que disponga de un intérprete de BASIC. Pero todavía se puede simplificar un poco más, como en la siguiente versión:



Obsérvese, en efecto, que la expresión J+4*(I−1) genera precisamente los números de 1 a 12 a medida que I varía de 1 a 3 y J varía de 1 a 4, de acuerdo con los límites de los bucles correspondientes. Todo esto quedará explicado con detalle más adelante.

En BASIC, para que aparezcan por la pantalla los valores de la tabla M, podemos hacer dos cosas: o bien escribirlos uno por uno, o bien conjuntamente, utilizando bucles. Por ejemplo, supongamos que queremos ver en la pantalla el valor del elemento de la tabla situado en la segunda fila y la tercera columna. Bastaría, para ello, con la instrucción siguiente:



y obtendremos el resultado (7). En cambio, para que la tabla nos aparezca entera, tendremos que utilizar el doble bucle siguiente:

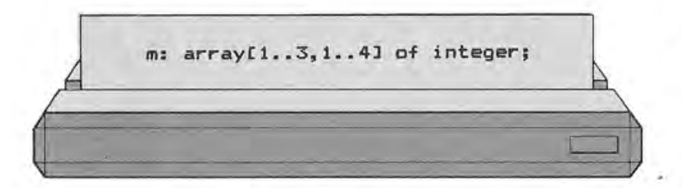
```
        FOR I=1 TO 3: FOR J=1 TO 4
        PRINT M(I,J);
        NEXT J: PRINT: NEXT I
 1  2  3  4
 5  6  7  8
 9  10  11  12
```

que producirá el resultado indicado. La primera instrucción PRINT (PRINT M (I,J);) termina en punto y coma para indicarle al intérprete que no debe escribir un paso de línea después de este elemento de la tabla. La segunda instrucción PRINT, que sólo se ejecuta una vez al final de cada fila, escribe precisamente el paso de línea correspondiente.

Finalmente, veamos un programa completo escrito en BASIC que declara una tabla, llena los valores correspondientes y los escribe por la pantalla:

```
10 DIM M(3,4)
20 FOR I=1 TO 3: FOR J=1 TO 4
30 LET M(I,J)=J+4*(I-1)
40 NEXT J: NEXT I
50 FOR I=1 TO 3: FOR J=1 TO 4
60 PRINT M(I,J);
70 NEXT J: PRINT: NEXT I
```

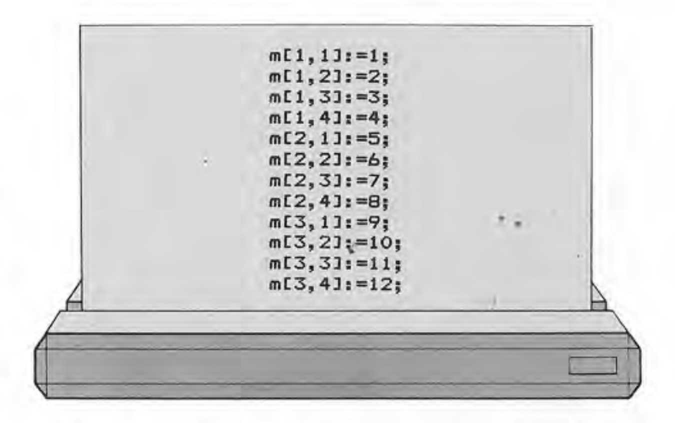
Veamos ahora cómo se trabaja con tablas en el lenguaje PASCAL. Como cualquier otra variable, es preciso declararlas, lo que se hace con una instrucción muy parecida a la que utilizamos en el capítulo anterior para declarar vectores:



```
m: array[1..3,1..4] of integer;
```

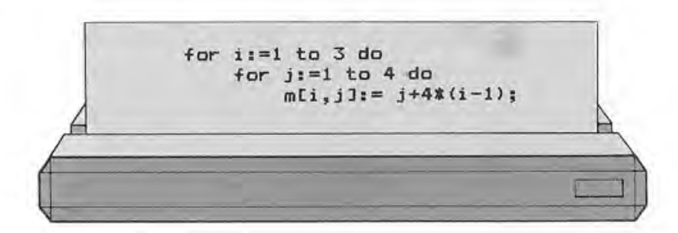
Esta instrucción define m como una tabla de números enteros («integer» en inglés) de tres filas y cuatro columnas. Obsérvese que se utiliza también la palabra reservada «array» (que significa «secuencia», «disposición», y en este caso equivale a «tabla», equivalente a DIM de BASIC.

En PASCAL también es posible llenar la tabla elemento a elemento, como vimos en BASIC. Se haría así:



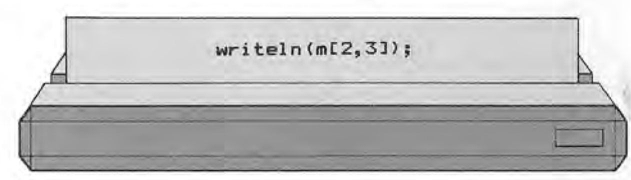
```
m[1,1]:=1;
m[1,2]:=2;
m[1,3]:=3;
m[1,4]:=4;
m[2,1]:=5;
m[2,2]:=6;
m[2,3]:=7;
m[2,4]:=8;
m[3,1]:=9;
m[3,2]:=10;
m[3,3]:=11;
m[3,4]:=12;
```

Pero en este caso es mejor hacerlo mediante un bucle parecido al del programa BASIC:



```
for i:=1 to 3 do
    for j:=1 to 4 do
        m[i,j]:= j+4*(i-1);
```

Para escribir un valor de la tabla en la pantalla, por ejemplo, el situado en la segunda fila y la tercera columna, podríamos utilizar la instrucción «write» (escribir, en inglés), equivalente a PRINT de BASIC, de la siguiente manera:



```
writeln(m[2,3]);
```

Para escribir la tabla entera, usaremos un doble bucle muy semejante al de BASIC, donde «write(m[i,j], ' ')» escribe el valor del elemento de la matriz m situado en la fila i y en la columna j, seguido de un blanco, pero sin pasar a la línea siguiente. En cambio, la instrucción «writeln», que se ejecuta al final de cada fila, escribe un paso de línea que nos permi-

te continuar escribiendo una nueva fila en la línea siguiente.

```
for i:=1 to 3 do
  begin
    for j:=1 to 4 do
        write(m[i,j],' ');
    writeln;
  end
```

Finalmente, veamos el programa completo escrito en PASCAL, que nuevamente declara una matriz llamada «m», llena sus casillas con los valores de 1 a 12 y los imprime en forma de tabla.

```
program TABLA;

var
  i, j: integer;
  m: array[1..3,1..4] of integer;

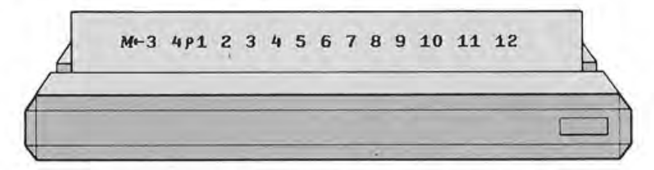
begin
  for i:=1 to 3 do
      for j:=1 to 4 do
          m[i,j]:= j+4*(i-1);
  for i:=1 to 3 do
      begin
          for j:=1 to 4 do
              write(m[i,j],' ');
          writeln;
      end
end.
```

que produce el siguiente resultado al ejecutarlo:

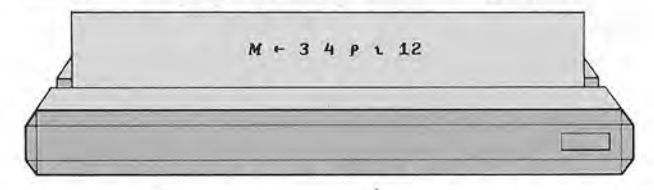
```
1 2 3 4
5 6 7 8
9 10 11 12
```

Por último, vamos a ver cómo se utilizan las tablas en el lenguaje APL. En este lenguaje no es necesario declararlas: basta con asignarle a una variable el valor correspondiente. Existen dos modos principales de construir tablas en el lenguaje APL: por enumeración o por cálculo.

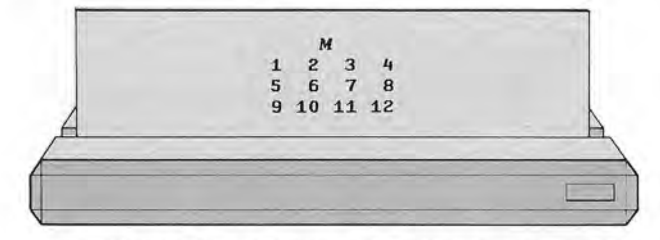
Una tabla se construye por enumeración por medio del símbolo ρ (la letra griega rho). A la izquierda de este símbolo se coloca el número de filas y el número de columnas que va a tener la matriz, y a su derecha se enumeran los valores que van a tomar los distintos elementos, leídos por filas. El resultado puede asignársele a una variable, que llamaremos M, como en los ejemplos anteriores:



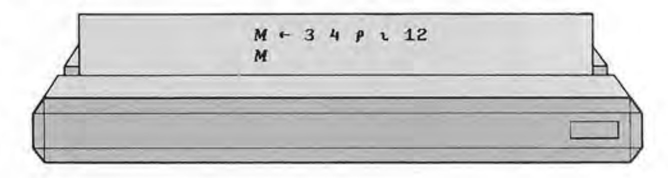
En este caso concreto, la expresión anterior puede reducirse, pues existe en APL un símbolo (representado por la letra griega iota) que, aplicado a cierto número entero positivo, produce la serie de todos los números sucesivos desde 1 hasta dicho entero. Por ejemplo, ι5 equivale a la serie 1, 2, 3, 4, 5. En nuestro caso, ι12 corresponderá a la serie de números del 1 al 12. Utilizando este símbolo, el programa anterior quedará:



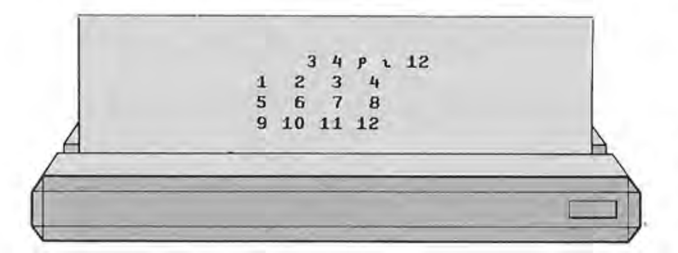
APL es un lenguaje especialmente diseñado para trabajar con facilidad con series y tablas de datos. Por esta razón, no son precisos bucles para presentar en la pantalla los valores de la tabla que acabamos de crear. Basta con invocar su nombre y obtendremos el resultado, correctamente encolumnado.



En consecuencia, el programa completo que crea una tabla de tres filas y cuatro columnas, llena sus valores y los imprime por la pantalla, se reduce en el caso de APL a las dos instrucciones siguientes:

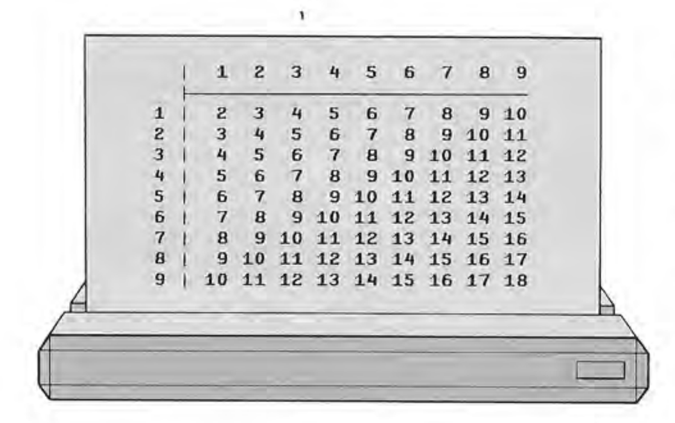


Esto en el caso de que queramos conservar la tabla en la variable M. Si lo único que queremos es verla, pero no guardarla, el programa anterior se reducirá a la instrucción única:



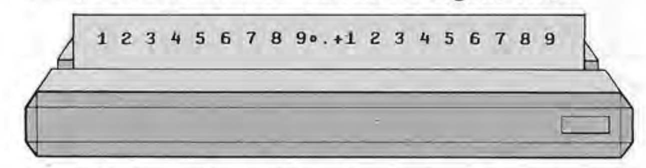
que, como se indica, produce directamente el resultado deseado.

La segunda manera de generar tablas en APL es aplicando un cálculo determinado a los elementos de dos series de datos. Recordemos, por un momento, la tabla de sumar que estudian los niños en el colegio, y que puede representarse así:

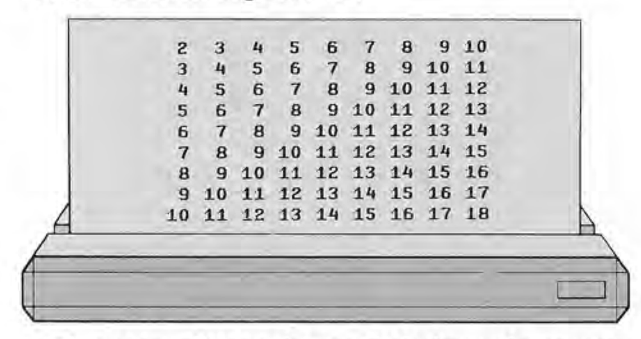


¿Cómo se obtiene esta tabla? Es fácil ver que se parte de dos series de datos iguales a 1, 2, 3, 4, 5, 6, 7, 8, 9, una de las cuales se encuentra en los encabezados de las filas y la otra en los de las columnas. Un elemento cualquiera de la tabla (por ejemplo, el de la tercera fila y la cuarta columna) se obtiene sumando el tercer valor de la serie de datos correspondiente a las filas (el encabezado de la tercera fila) con el cuarto valor de la correspondiente a las columnas (el encabezado de la cuarta columna). En definitiva, operando de esta manera con todas las combinaciones posibles de un elemento de la serie de filas y uno de la serie de columnas, se obtiene la tabla entera.

En APL esta operación se representa, simplemente, con la línea siguiente:

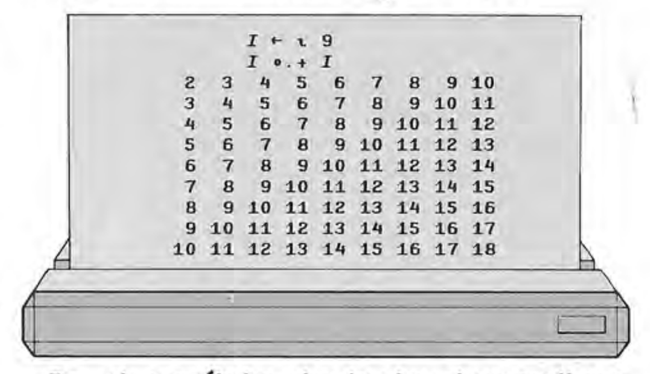


donde la colección de datos correspondiente a las filas se escribe a la izquierda; la de las columnas a la derecha; y en medio se coloca un círculo pequeñito (que suele obtenerse en los teclados APL de la tecla de la J en posición de mayúsculas), seguido de un punto y de la operación que se desea realizar entre los elementos de las dos colecciones de datos. El resultado de ejecutar la línea anterior será el siguiente:



El procedimiento es totalmente general. Las series izquierda y derecha no tienen por qué ser iguales, y en lugar de la operación suma podríamos haber utilizado otra cualquiera. Por ejemplo, si pusiéramos el signo «por» (×), obtendríamos la tabla de multiplicar.

Recordando que el símbolo ι aplicado a un entero nos produce los números consecutivos desde 1 hasta el entero, podemos simplificar más aún la generación de la tabla de sumar, reduciéndola a:

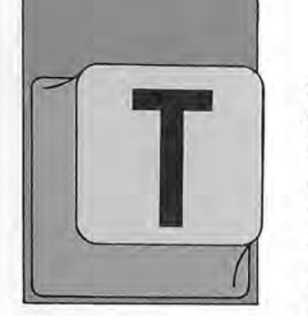


En el capítulo siguiente desarrollaremos una aplicación completa de utilidad práctica, basada en la utilización de tablas de valores.



APLICACIONES

TRATAMIENTO DE TEXTOS



TODOS hemos oído hablar de los Tratamientos de Textos como una de las herramientas más necesarias y útiles disponibles en un ordenador. Pero ¿sabemos realmente qué es y para qué sirve un Tratamiento de Textos?

Una aplicación de este tipo es algo más que un programa para escribir documentos, es decir, no es una "máquina de escribir electrónica", sino que es un eslabón más de la cadena de utilización de un ordenador como herramienta de productividad, ya que permite la introducción de información en el ordenador para que sea elaborada más adelante; ya de forma impresa o bien por otro programa, como una base de datos.

Aparte del hecho de introducir el texto o datos en el ordenador, podemos resaltar algunos de los aspectos más relevantes que lo diferencian de una máquina de escribir:

— Corrección de una parte del texto sin necesidad de recomponer o corregir el resto del texto, ya que éste se reajustará de forma automática al introducir la corrección.

— Posibilidad de realizar "plantillas" que permiten evitar la introducción de un texto que tenga que ser repetido en varios documentos, como cabeceras, despedidas, dirección del usuario, etc.

— El almacenamiento se reduce de una forma considerable, ya que podemos tener una gran cantidad de documentos archivados en un solo disquete.

Antes de pasar a detallar las características básicas de un procesador de texto, es conveniente hacer notar que su uso de una forma eficaz está condicionado al ordenador disponible, siendo preferible un ordenador con una capacidad de memoria algo elevada (128 K min.), y la utilización de disquetes como unidad de almacenamiento externa.



Características técnicas

— *Avance automático de línea*

El usuario no tiene que preocuparse de dónde acaba la línea actual del texto,

ya que es el programa el encargado de controlar la longitud de la línea y pasar a la siguiente si se excede ésta.

— *Inserción y supresión de caracteres* en las posiciones deseadas.

— *Ajuste automático de los párrafos,* en función de las modificaciones realizadas en él, en alguna de las funciones anteriormente dichas.

13 de Febrero de 1987

Sr. D. E. Gurbindo
Príncipe de Vergara, 291
28016 Madrid

Estimado Eduardo:

A raíz de la conversación que mantuvimos el
pasado 26 de los corrientes, considero oportu-
no el mantener una reunión el próximo día 28,
con el objeto de discutir los pormenores de la
negociación.

Me congratulo con la noticia del nacimiento de
su segundo vástago, aprovechando la ocasión
para expresarle mi enhorabuena a su encan-
tadora esposa.

Afectuosamente:

Virginia Muñoz

Párrafo sin sangrado.

13 de Febrero de 1987

Sr. D. E. Gurbindo
Príncipe de Vergara, 291
28016 Madrid

Estimado Eduardo:

A raíz de la conversación que mantuvimos el
pasado 26 de los corrientes, considero oportu-
no el mantener una reunión el próximo día 28,
con el objeto de discutir los pormenores de la
negociación.

Me congratulo con la noticia del nacimiento
de su segundo vástago, aprovechando la oca-
sión para expresarle mi enhorabuena a su en-
cantadora esposa.

Afectuosamente:

Virginia Muñoz



Párrafo reformado.

— *Justificación del texto* en los márgenes escogidos por el usuario. Con esta función el texto se podrá alinear en alguno de los márgenes, izquierdo o derecho, o en ambos, mejorando con ello la presentación del texto.

13 de Febrero de 1987

Sr. D. E. Gurbindo
Príncipe de Vergara, 291
28016 Madrid

Estimado Eduardo:

A raíz de la conversación que mantuvimos el
pasado 26 de los corrientes, considero oportu-
no el mantener una reunión el próximo día 28,
con el objeto de discutir los pormenores de la
negociación.

Me congratulo con la noticia del nacimiento
de su segundo vástago, aprovechando la oca-
sión para expresarle mi enhorabuena a su en-
cantadora esposa.

Afectuosamente:

Virginia Muñoz

Texto ajustado a la derecha y a la izquierda.

13 de Febrero de 1987

Sr. D. E. Gurbindo
Príncipe de Vergara, 291
28016 Madrid

Estimado Eduardo:

A raíz de la conversación que mantuvimos
el pasado 26 de los corrientes, considero
oportuno el mantener una reunión el próximo
día 28, con el objeto de discutir los
pormenores de la negociación.

Me congratulo con la noticia del nacimiento
de tu segundo vástago, aprovechando la
ocasión para expresarle mi enhorabuena a su
encantadora esposa.

Afectuosamente:

Virginia Muñoz



Texto ajustado a la izquierda.

— *Búsqueda y sustitución* de palabras y frases, de forma completamente automática.

APLICACIONES



Estimado Roberto:
El motivo de la presente es convocarte para la concentración preparatoria del partido de fútbol que tendrá lugar frente a la selección de Madagascar.

Te recomiendo Roberto que te lleves puestas las botas por lo que pudiera pasar. Sin más que relatarte, Roberto, recibe un cordial saludo de tu presidente:

Paloma Gandiel



Ejemplo de búsqueda y sustitución.

13 de Febrero de 1987

Sr. D. E. Gurbindo
Príncipe de Vergara, 291
28016 Madrid

Estimado Eduardo:

A raíz de la conversación que mantuvimos el pasado 26 de los corrientes, considero oportuno el mantener una reunión el próximo día 28, con el objeto de discutir los pormenores de la negociación.

Me congratulo con la noticia del nacimiento de su segundo vástago, aprovechando la ocasión para expresarle mi enhorabuena a su encantadora esposa.

Afectuosamente:

Virginia Muñoz



Espaciado sencillo entre líneas.

Estimado Gaby:
El motivo de la presente es convocarte para la concentración preparatoria del partido de fútbol que tendrá lugar frente a la selección de Madagascar.

Te recomiendo Gaby que te lleves puestas las botas por lo que pudiera pasar. Sin más que relatarte, Gaby recibe un cordial saludo de tu presidente:

Paloma Gandiel



Ejemplo de búsqueda y sustitución.

13 de Febrero de 1987

Sr. D. E. Gurbindo

Príncipe de Vergara, 291

28016 Madrid

Estimado Eduardo:

A raíz de la conversación que mantuvimos el pasado 26 de los corrientes, considero oportuno el mantener una reunión el próximo día 28, con el objeto de discutir los pormenores de la negociación.

Me congratulo con la noticia del nacimiento de tu segundo vástago, aprovechando la ocasión para expresarle mi enhorabuena a su encantadora esposa.



Espaciado doble entre líneas.

— *Paginación automática del texto,* ajustando el texto entre los márgenes izquierdo y derecho, superior e inferior en cada página.

— *Ajuste del salto entre líneas*

— *Impresión del texto* con posibilidad de elegir el estilo de impresión de todo o de una parte del texto.

— *Numeración automática* de las páginas.

— **Movimiento** de parte del texto de una posición a otra o posibilidad de almacenar éste en el disco para su posterior recuperación.

— *Marginación variable.* Esta función permite ajustar los márgenes del documento de tal forma que se acomoden al tamaño de la página que vamos a usar.

13 de Febrero de 1987

Sr. D. E. Gurbindo
Príncipe de Vergara, 291
28016 Madrid

Estimado Eduardo:

A raíz de la conversación que mantuvimos el pasado 26 de los corrientes, considero oportuno el mantener una reunión el próximo día 28, con el objeto de discutir los pormenores de la negociación.

Me congratulo con la noticia del nacimiento de tu segundo vástago, aprovechando la ocasión para expresarle mi enhorabuena a su encantadora esposa.

Afectuosamente:

Virginia Muñoz


Margen izquierdo estrecho.

13 de Febrero de 1987

Sr. D. E. Gurbindo
Príncipe de Vergara, 291
28016 Madrid

Estimado Eduardo:

A raíz de la conversación que mantuvimos el pasado 26 de los corrientes, considero oportuno el mantener una reunión el próximo día 28, con el objeto de discutir los pormenores de la negociación.

Me congratulo con la noticia del nacimiento de tu segundo vástago, aprovechando la ocasión para expresarle mi enhorabuena a su encantadora esposa.


Margen izquierdo ancho.



Funciones avanzadas

— Creación automática de índices. Con esta posibilidad podremos realizar un extracto o un sumario del texto.

— Inserción automática de notas a pie de página y cabeceras.

— Escritura en varias columnas.

— Posibilidad de realización de documentos personalizados con la ayuda de un fichero de datos.

— Corrección ortográfica automática.

— Inclusión de gráficos en el texto.

Actualmente están apareciendo en el mercado, además de tratamientos de textos tradicionales, una serie de programas que permiten realizar todas las funciones necesarias para la edición y composición de textos, logrando unos documentos con una calidad muy similar a la de los creados en una imprenta.



Impresoras

Un punto importante a tener en cuenta a la hora de trabajar con un procesador de textos es la calidad y el tipo de la impresora disponible, ya que el resultado final del documento depende en gran parte de la calidad de la impresora y de sus posibilidades.

PASCAL

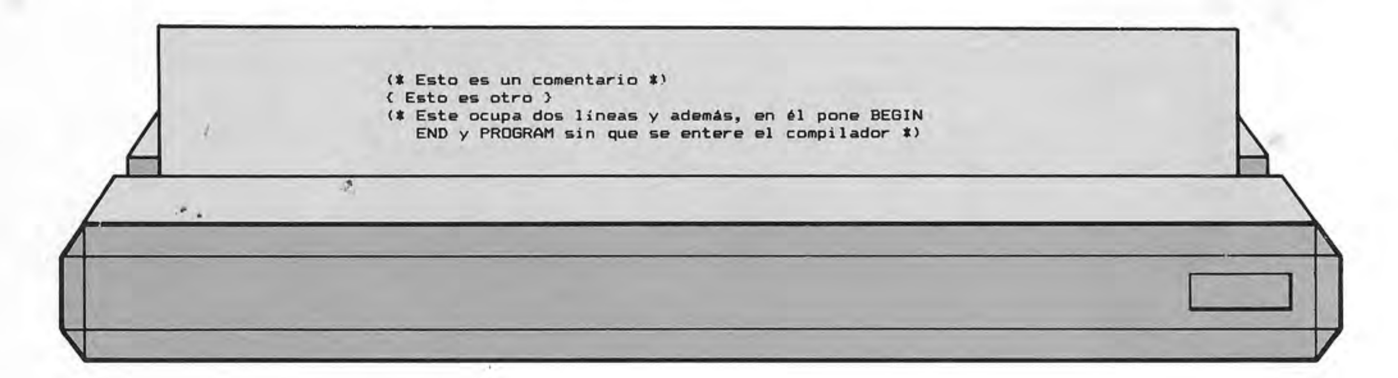


Comentarios de un programa

PARA que un programa resulte más fácil de entender, sería conveniente poner notas explicativas de las diferentes partes de que consta. Estas notas son lo que se denomina comentarios, y todos los lenguajes de programación permiten añadirlos a los programas de una manera u otra.

En PASCAL se pueden poner comentarios en CUALQUIER sitio en que pueda aparecer un espacio en blanco (excepto en medio de un texto), pues a la hora de traducir el programa, el compilador pasa por encima de ellos; sencillamente, ignora su existencia.

El comienzo de un comentario se indica con un paréntesis izquierdo inmediatamente seguido de un asterisco (o bien una llave izquierda si nuestro ordenador tiene ese símbolo); a partir de ahí, el compilador considera que todo es comentario hasta que se encuentra un asterisco seguido de un paréntesis derecho (o bien una llave derecha). Un comentario puede ocupar todas las líneas que se deseen y dentro de él se puede escribir CUALQUIER cosa. Posibles comentarios serían, pues:

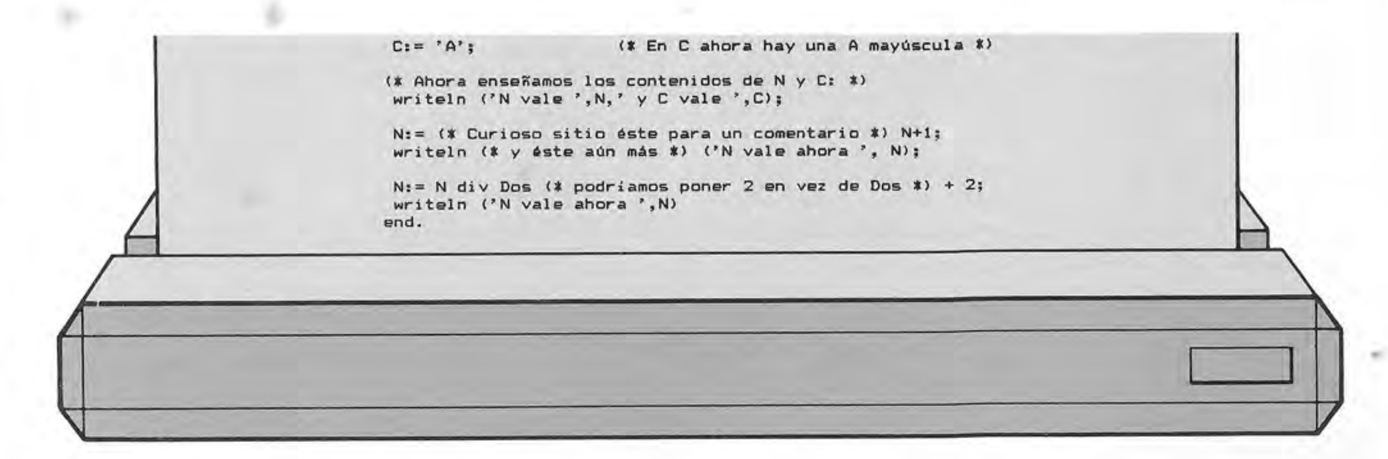


```
(* Esto es un comentario *)
{ Esto es otro }
(* Este ocupa dos líneas y además, en él pone BEGIN
   END y PROGRAM sin que se entere el compilador *)
```

Vamos a escribir ahora el programa ASIGNACION que hicimos anteriormente con comentarios:

```
program Asignacion;
    (* Este programa consta de una serie de ejemplos
       de asignación de datos a variables  *)
const
  Dos = 2;
var
  N: integer;  (* En esta variable se guardarán números *)
  C: char;     (* y en ésta letras *)

begin
 N:=  7 ;              (* Tras esto, en N hay un siete   *)
```



```
 C:= 'A';                (* En C ahora hay una A mayúscula *)

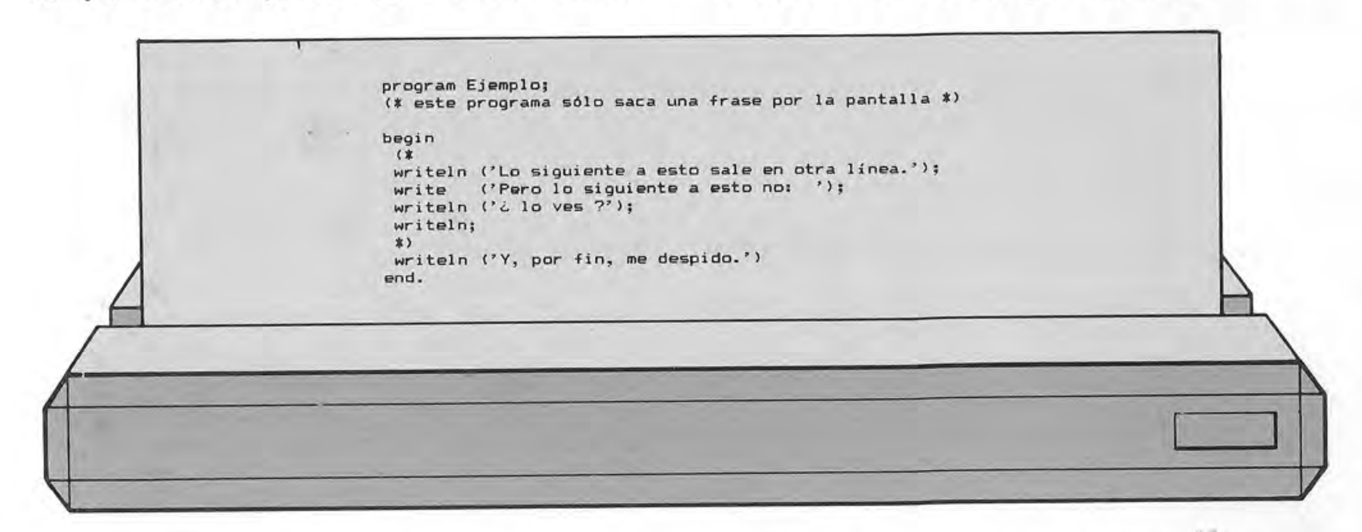
(* Ahora enseñamos los contenidos de N y C: *)
 writeln ('N vale ',N,' y C vale ',C);

 N:= (* Curioso sitio éste para un comentario *) N+1;
 writeln (* y éste aún más *) ('N vale ahora ', N);

 N:= N div Dos (* podríamos poner 2 en vez de Dos *) + 2;
 writeln ('N vale ahora ',N)
end.
```

El propósito de los comentarios es hacer los programas más claros, y por ello conviene ponerlos de manera que se distingan de lo demás a primera vista; en el ejemplo está claro que hay mejores sitios para algunos de ellos.

Cuando se está desarrollando un programa, a menudo sucede que hay partes ya escritas que no nos interesa utilizar por el momento. Una solución sería eliminarlas del programa, pero entonces habría que volver a escribirlas cuando se necesitasen de nuevo. Hay una posibilidad mucho más interesante: si delante y detrás de la zona a descartar ponemos, respectivamente, (* y *), el compilador creerá que todo es un comentario y pasará por encima olímpicamente:



```
program Ejemplo;
(* este programa sólo saca una frase por la pantalla *)

begin
  (*
 writeln ('Lo siguiente a esto sale en otra línea.');
 write   ('Pero lo siguiente a esto no:  ');
 writeln ('¿ lo ves ?');
 writeln;
  *)
 writeln ('Y, por fin, me despido.')
end.
```

(Hay una situación curiosa que muy raramente se da: si en lugar de '¿lo ves?' estuviera, por ejemplo, 'Mira estos dos símbolos: *)', al llegar ahí, el compilador creería que acaba el comentario, con lo que al pasar a lo siguiente se produciría un error.)

Indentación

A lo largo de todos los ejemplos vistos hasta ahora hemos utilizado la denominada «indentación», que consiste en utilizar márgenes izquierdos variables para resaltar las diferentes partes de un programa. En el programa ASIGNACION, por ejemplo, al declarar las variables, sus nombres aparecen encolumnados y un poco más a la derecha de la palabra VAR que marca el inicio de esa zona; igualmente, BEGIN y END están encolumnadas, pero las instrucciones que enmarcan están todas un poco más a la derecha.

Por supuesto, no es obligatorio escribir así los programas, pero es muy conveniente. Todos los programas que iremos escribiendo a lo largo de la colección harán un empleo aún mayor de la indentación.



Tipos de datos básicos

Hasta ahora hemos visto programas manejando básicamente dos tipos de datos: números y textos o caracteres sueltos. También hemos aprendido a escribir programas muy sencillos, que siempre constan de una secuencia de instrucciones que se ejecutan una detrás de otra, una única vez. Antes de empezar a estudiar cómo hacer programas más complejos, vamos a ver en detalle los tipos de datos que ya conocemos y uno nuevo, el tipo BOOLEAN.



El tipo Integer

Este tipo es el que corresponde a los números enteros. Ya hemos aprendido a:

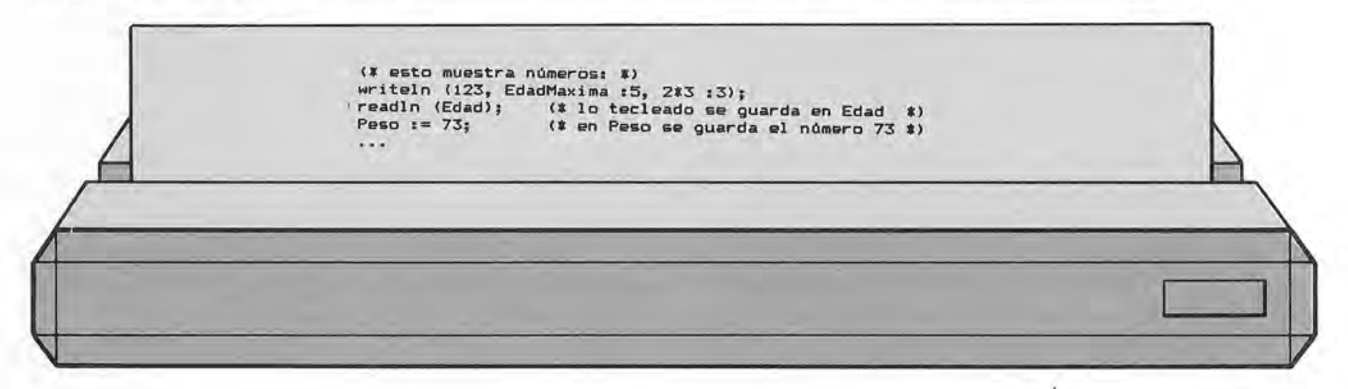
— escribir constantes: 123, –55, 0, 1987

— declararlas: const EdadMaxima = 110;

— definir variables: var Edad: integer; Dia: 1..31;

— escribir expresiones: (EdadMaxima – Edad) * 100

También sabemos cómo mostrar valores de tipo INTEGER, leerlos desde teclado y guardarlos en variables:



Los operadores permitidos son: +, –, *, DIV y MOD. Hay dos funciones importantes disponibles para este tipo de datos:

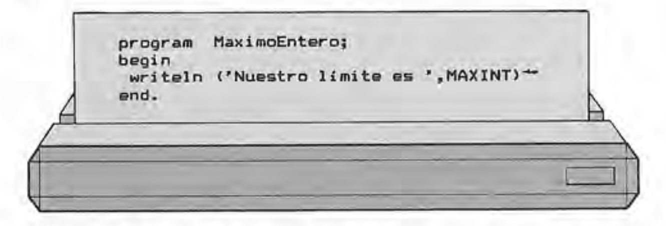
— Si escribimos ABS (X), esto da como resultado el valor absoluto de X, o sea, X mismo si es positivo, o –X si es negativo (X puede ser cualquier constante, variable o expresión de tipo INTEGER): ABS (–5) equivale a 5, al igual que ABS (7 – 2).

— SQR (X) da como resultado el cuadrado de X, X por X (SQR es la contracción de SQUARE, cuadrado en inglés): SQR (2 + 2) es 16 y SQR (–7) es 49. Estas funciones se pueden utilizar como parte de expresiones con total libertad, y equivalen a poner en su lugar el resultado que fueran a dar en el momento de evaluar la expresión.

Los valores que se pueden manejar con constante, variables y expresiones enteras tienen unos límites que dependen del ordenador y del compilador que tengamos. Para conocer estos límites existe una constante predeclarada (es decir, que se puede utilizar aunque no la hayamos declarado nosotros) cuyo nombre es MAXINT.

Por ello, si el valor de MAXINT («máximo entero») fuese 32767, que es el valor más corriente con la mayoría de compiladores para ordenadores personales, los valores de tipo INTEGER deben estar entre –32767 y +32767, ambos inclusive. Así, la expresión 1000 * 1000, por ejemplo, daría un resultado que se sale de los límites provocando o bien la parada del programa o bien un resultado erróneo, según el compilador utilizado. Igualmente, la expresión 1000 * 1000 div 10000, aunque da un resultado aceptable (100), como se hace primero la multiplicación, producirá una situación similar al ser calculada.

Para conocer nuestro límite particular, podríamos emplear el siguiente programa:



Para cuando se tengan que manejar números que, o bien no son enteros, o

bien se salen de los límites, existe el tipo REAL, que veremos en otra ocasión.



El tipo Char

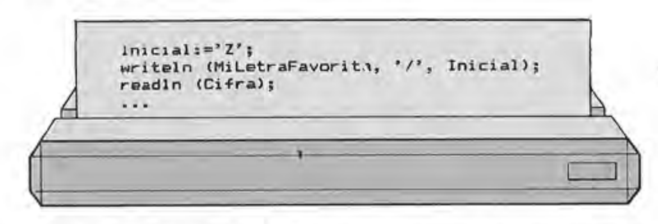
Este tipo es el utilizado para manejar caracteres sueltos, es decir, una letra, una cifra, un signo de puntuación, un espacio en blanco, etc. Al igual que con el tipo INTEGER ya sabemos:

— escribir constantes: 'A', 'O', '/', ' . '.

— declararlas: const MiLetraFavorita = 'P';

— definir variables: var Inicial: char: Cifra: '0'..'9';

así como mostrar y leer de teclado valores de tipo CHAR y guardarlos en variables:



```
Inicial:='Z';
writeln (MiLetraFavorita, '/', Inicial);
readln (Cifra);
...
```

El conjunto de caracteres disponibles depende del ordenador, pero SIEMPRE existen, al menos, las 26 mayúsculas del alfabeto inglés:

A, B, C, D, E, F, G, H, I, J, K, L, M,
N, O, P, Q, R, S, T, U, V, W, X, Y, Z.

las nueve cifras decimales:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

y el espacio en blanco.



Ordinales

Internamente, y sin que nosotros nos demos cuenta de ello, los caracteres se guardan en memoria utilizando números, uno distinto para cada posible valor. Son lo que se denomina NUMEROS ORDINALES del conjunto de caracteres.

Las letras están ordenadas según estos números. Si, por ejemplo, el ordinal de la letra A fuera el 65, el de la B sería el 66 y así hasta la Z. Lo mismo sucede con las cifras del 0 al 9.

Si se quisieran utilizar estos números en expresiones de tipo entero, la función ORD (C), donde C es un valor cualquiera de tipo CHAR, proporciona un resultado de tipo INTEGER igual al ordinal del carácter. A la inversa, si X fuera un valor de tipo INTEGER, CHR(X) nos proporcionaría el carácter cuyo ordinal es X.

Utilizando la función ORD, vamos a escribir un programa que nos sirva para conocer los ordinales de algunos caracteres de nuestro ordenador:

```
program SacaOrdinales;
(* esta constante de texto es para escribir
menos: *) const Txt = 'El ordinal del carácter';

begin
 writeln (Txt, 'A', ' es ', ord ('A'));
 writeln (Txt, 'a', ' es ', ord ('a'));
 writeln (Txt, 'O', ' es ', ord ('O'));
 writeln (Txt, 'espacio en blanco es ',
 ord (' '));
  (* poner aquí todos las instrucciones
  que se quieran: *) end.
```

Existen además otras dos funciones:

— PRED (C), donde C es un carácter, nos devuelve el carácter anterior a él. PRED ('Y') equivale, pues, a poner 'X'.

— SUCC (C) devuelve el siguiente a C. (El lector atento se dará cuenta en seguida de que PRED(C) equivale a poner CHR(ORD(C)−1) y SUCC(C) a CHR(ORD(C)+1.)

Normalmente, en los ordenadores personales se utilizan los códigos ASCII (siglas de American Standard Code for Information Interchange, pronunciado «aski») en que las letras mayúsculas empiezan por el ordinal 65, las cifras por el 48 y el espacio en blanco, por ejemplo tiene el 32.

Por ahora, en lo referente a textos con más de un carácter, lo único que sabemos hacer es escribirlos y declararlos como constantes. Por supuesto, es posible leerlos de teclado y utilizar variables para guardarlos, pero el tipo especial de dato necesario para ello se verá en otra ocasión.



El tipo Boolean

Si queremos que nuestros programas lleguen a ser más útiles de lo que son los que hemos escrito hasta el momento, está claro que deben ser capaces de to-

mar decisiones en función de determinados resultados. Por ejemplo:

«Si la edad es superior a 17 años, entonces hacer tal cosa».

Cuando el ordenador se encontrase con que tiene que tomar una decisión semejante, la primera operación que debería efectuar sería la de comparar la edad (que podría ser, por ejemplo, la contenida en una variable de tipo INTEGER) con 17 para ver si es mayor o no. Si fuese mayor, el resultado de la operación «Edad es superior a 17» sería CIERTO y en caso contrario sería FALSO.

Las expresiones de este tipo, que pueden dar como resultado CIERTO o FALSO, se llaman expresiones LOGICAS o de tipo BOOLEAN.

En PASCAL existe el tipo de dato BOOLEAN, que es aquél que sólo puede tomar uno de entre dos posibles valores, CIERTO y FALSO, valores que se representan mediante las constantes predefinidas TRUE (cierto en inglés) y FALSE.

— Al igual que sucede con los otros tipos que ya conocemos, podemos tener variables BOOLEAN donde guardar estos datos y que, por tanto, sólo pueden tomar los valores TRUE y FALSE:

```
var MayorDeEdad : boolean;
```

— Se pueden asignar valores a esas variables:

```
MayorDeEdad := true;
```

— Se pueden mostrar valores BOOLEAN:

```
writeln (MayorDeEdad);
```

(Esto último haría que se escribiera la palabra TRUE o FALSE, según el valor de la variable.)

— Sin embargo, NO se puede utilizar READ o READLN con variables BOOLEAN.



Comparaciones entre números

Muy frecuentemente se desean comparar números enteros para tomar decisiones, como en el caso de la edad. Para ello se utilizan expresiones lógicas de comparación que pueden dar como resultado el valor lógico TRUE o FALSE, según el caso. Estas expresiones siempre constan de los dos valores a comparar (constantes, variables o expresiones enteras) separados entre sí por un operador de comparación:

```
Edad > 17
```

Esta expresión debe leerse como «Edad mayor que 17», y daría TRUE o FALSE como resultado según que EDAD fuese mayor o no que 17, respectivamente. Podríamos poner, pues:

```
MayorDeEdad := (Edad > 17);
```

Además del operador de comparación «mayor que», cuyo símbolo es >, existen los siguientes:

- \>= significa «mayor o igual que»
- < significa «menor que»
- <= significa «igual o menor que»
- = significa «igual a»
- <> significa «distinto de»

Además, la función ODD (X) (odd significa impar entre otras cosas), donde X es un valor entero, devuelve el valor TRUE si X es impar y FALSE en caso contrario.

Vamos a escribir un programa que guarde en una variable la edad que tecleemos y nos diga a continuación si corresponde a un mayor de edad y si es una edad impar:

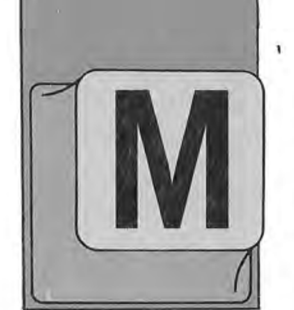
```
program MiraEdad;
const
 LimiteMenores = 17;
var
 Edad         : integer;
 MayorDeEdad : boolean;
begin
 write  ('Por favor, teclee su edad: ');
 readln (Edad);
 MayorDeEdad := (Edad > LimiteMenores);

  (* ahora sacamos las respuestas con TRUE O
FALSE: *) writeln ('¿Es mayor de edad?  ',
MayorDeEdad); writeln ('¿Es una edad impar? ',
odd (Edad)) end.
```

OTROS LENGUAJES

SISTEMAS OPERATIVOS MS/DOS (2)

Redirección de la entrada y la salida



M S/DOS permite dirigir o redireccionar la entrada o la salida de un programa, de manera que ésta se produzca donde desee el usuario, que bien puede ser a un fichero o a un dispositivo periférico. Por ejemplo, se puede redireccionar la salida de un comando para que el resultado, en vez de obtenerse por pantalla, aparezca por la impresora, o redirigir la entrada de un programa para que obtenga datos de un fichero en disco en vez del teclado.

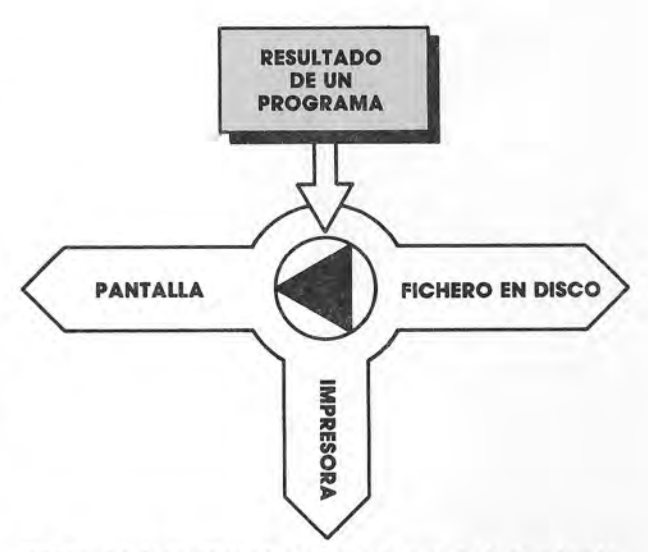
La redirección se especifica con los signos "<" para la entrada, y ">" para la salida. Por ejemplo, un comando con la sintaxis:

orden > fichero

estaría dirigiendo su salida hacia el fichero, cuyo nombre se especifica. De manera análoga se podría hacer:

orden < fichero

para redireccionar la entrada.

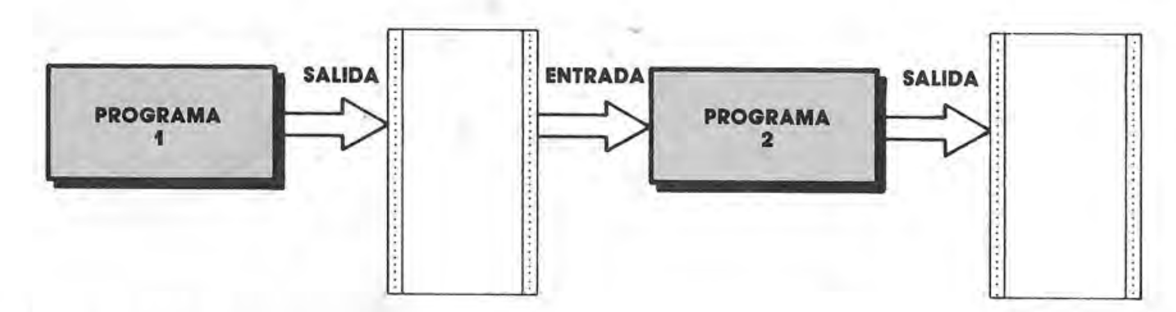


En MS/DOS es posible redireccionar tanto la salida como la entrada, ya sea a ficheros en disco como a dispositivos periféricos.



Conexión de Programas mediante «PIPES»

Otra característica que posee MS/DOS es la de poder conectar programas mediante un mecanismo llamado "pipe-line" (tubería). La "conexión" consiste en que los resultados de un programa pueden ser tomados por un segundo como datos de entrada, y la salida de éste puede ser a su vez la entrada de un tercero, y así sucesivamente. El operador utilizado para esto es el símbolo ":", y el formato de dos programas conectados por "pipes" es: Programa 1: Programa 2.



En MS/DOS los programas pueden conectarse de manera que la salida de uno sea la entrada de otro.

Es importante resaltar el hecho de que el primer programa debe generar datos capaces de "alimentar" al programa que viene a continuación, el cual debe "consumirlos" y generar nuevos datos para que sean utilizados por el programa siguiente (si hubiese), y así hasta finalizar la cadena.

Comandos fundamentales de MS/DOS

MS/DOS tiene un intérprete de comandos (el fichero COMMAND.COM), que espera a que tecleemos cualquier orden. Si es correcta, se encarga de procesarla, y si no lo es devolverá un mensaje de error. Dentro de los comandos u órdenes que puede procesar este intérprete hay que distinguir dos tipos:

— Comandos que están siempre en memoria RAM (internos).
— Comandos que están almacenados en disco (externos).

Si empleamos una orden interna, COMMAND.COM no tiene más que mirar en la memoria RAM del ordenador y ejecutarla. Si, por el contrario, hacemos referencia a una orden externa, se provoca una búsqueda en el disco del fichero correspondiente a ese comando.

Dentro de todas las órdenes de MS/DOS, hemos seleccionado las más importantes, las cuales hemos englobado en cuatro grupos:

— Comandos de tratamiento y mantenimiento de discos.
— Comandos para operaciones con ficheros.
— Comandos de salida de ficheros.
— Comandos de manejo de la estructura de ficheros.



Comandos de tratamiento y mantenimiento de discos

DIR

Produce un listado de los ficheros y/o subdirectorios del directorio especificado. La información de este listado incluye:

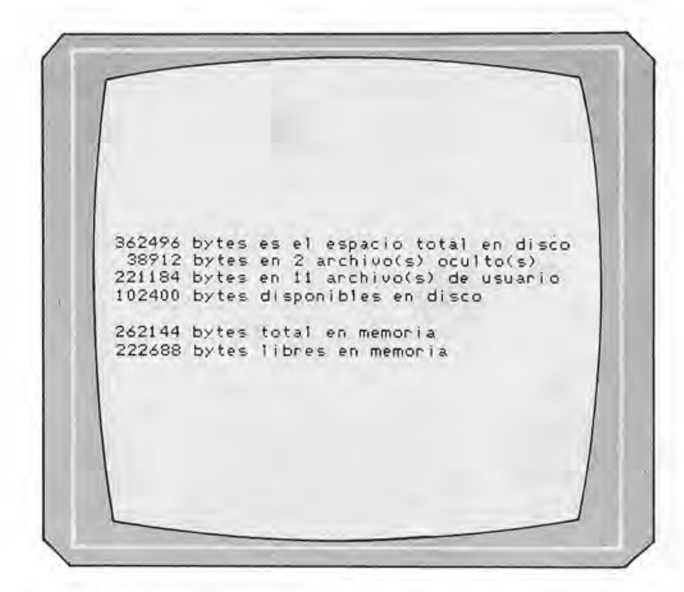
— Nombre y extensión del fichero.
— Tamaño del fichero (en bytes).
— Fecha y hora de la última actualización del fichero (si nunca se ha modificado, fecha y hora de su creación).
— Etiqueta del volumen del disco.
— Número total de ficheros.
— Espacio libre en disco.

Asimismo, puede obtenerse esta información para un solo fichero, o para un grupo de ficheros especificados.

CHKDSK

Comprueba la situación de un disco en lo referente a espacio ocupado y posibles errores. En su opción normal de trabajo, CHKDSK muestra la siguiente información:

— Espacio total en disco y en memoria RAM.
— Espacio ocupado en disco por los ficheros del sistema (ocultos al trabajo de DIR).
— Espacio ocupado en disco por los ficheros de usuario.
— Espacio libre en disco y en memoria.





Ejecución del comando CHKDSK.

En cuanto a los errores, CHKDSK informa de lo siguiente:

— Daños internos en el disco, como, por ejemplo, encontrar sectores o pistas inservibles.

— Bloques de información no ligados a ningún fichero dentro del directorio.

— Almacenamiento de los ficheros en sectores no contiguos.

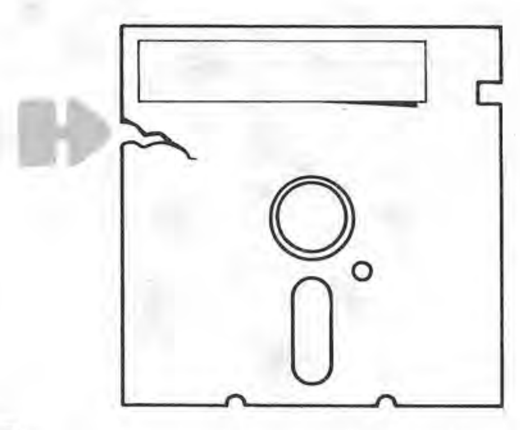
Realmente esto no es un error, pero hace que el acceso a los ficheros se haga más lento. Por tanto, si hay muchos en este estado, es conveniente pasarlos a otro disco.

FORMAT

El comando FORMAT es el encargado de preparar los discos vírgenes, para su posterior utilización como almacenamiento secundario de datos y programas.

Durante el formateo, el ordenador examina el disquete, indicando los posibles defectos que éste pudiera tener. En caso de existir alguna anomalía, FORMAT hace dos cosas:

— Advertir al usuario de la presencia de los defectos, indicando el tamaño en bytes de las zonas defectuosas, así como el espacio utilizable.



Una de las funciones del comando FORMAT es marcar las áreas defectuosas de los disquetes.

— Marcar el área defectuosa de manera que ésta no sea utilizable por el sistema operativo, aunque sí el resto del disquete.

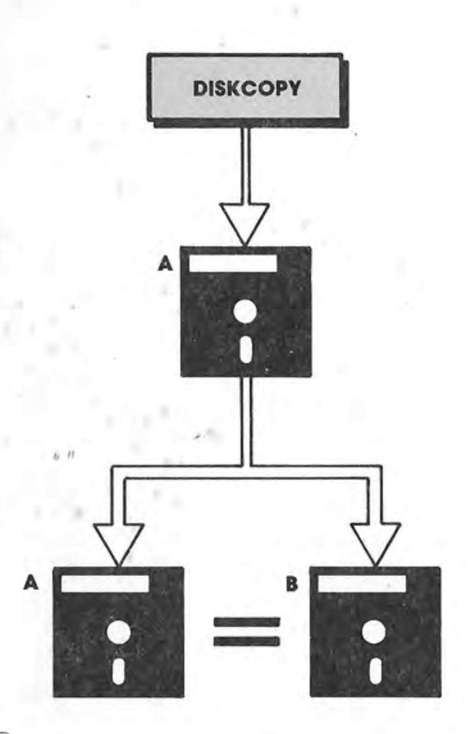
Como casi todos los comandos, FORMAT tiene una serie de opciones que diversifican sus funciones. Una de las más importantes es la que permite incluir los ficheros del sistema operativo en el disquete a formatear, con lo cual se puede emplear para inicializar el sistema.

SYS

El comando SYS permite cargar los ficheros del sistema operativo y las pistas de inicialización en un disquete previamente formateado. La diferencia que hay entre esta orden y el comando FORMAT con la opción antes mencionada es que SYS permite realizar esta operación en los discos protegidos, cosa que FORMAT no hace.

DISKCOPY

Con el comando DISKCOPY se copia el contenido de un disquete en otro disquete, de forma que se obtiene una reproducción exacta del original. En casi todas las versiones de MS/DOS, DISKCOPY formatea el disco destino en caso de que no lo esté. Además, no admite disquetes con defectos (aquéllos que detecta FORMAT), ni tampoco optimiza los resultados del disco, es decir, si en el disco origen estaban los ficheros distribuidos esporádicamente, en el disco destino se copiarán con esa misma distribución.

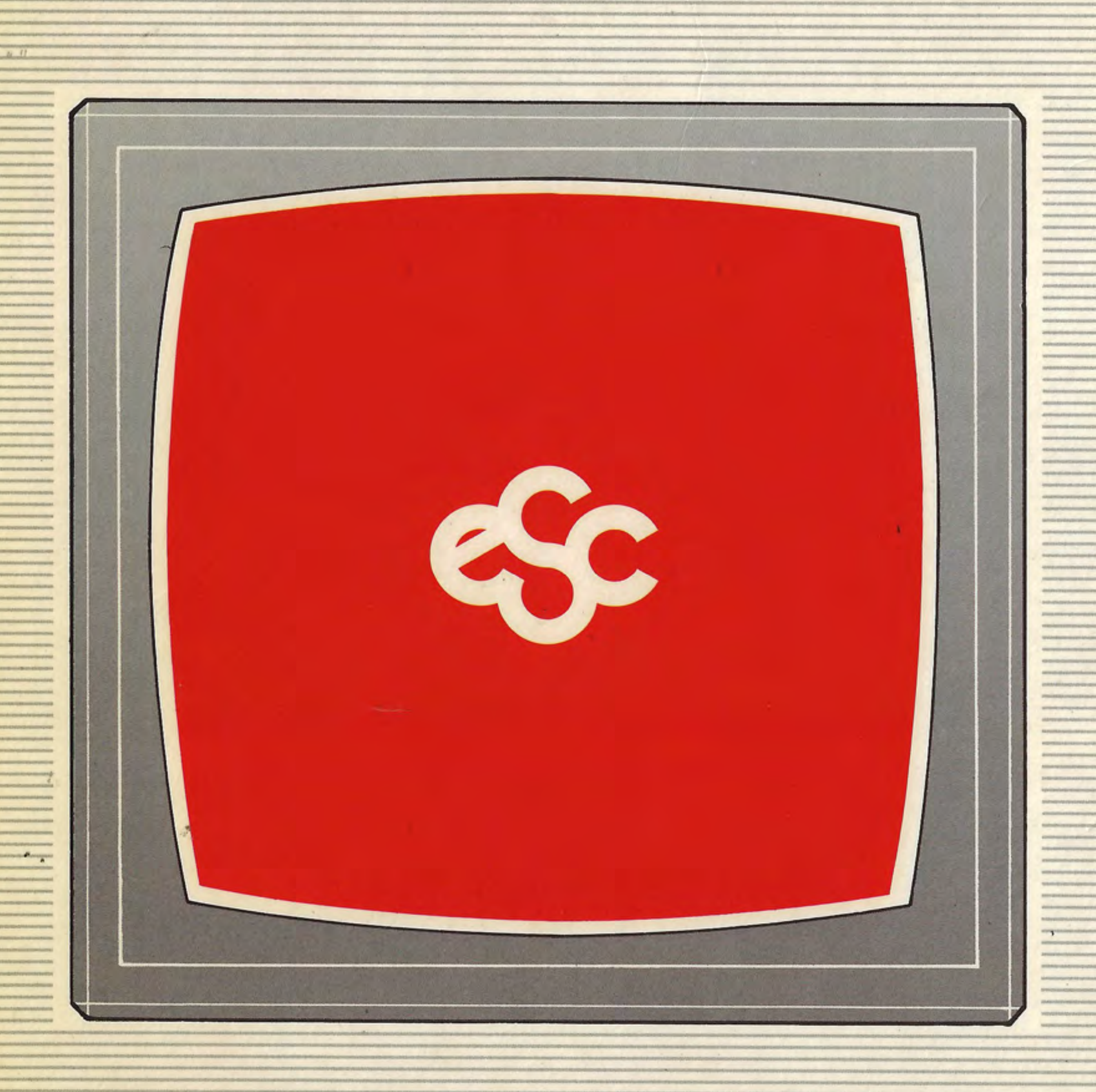


El comando DISKCOPY reproduce de manera exacta el contenido de un disquete.

DISKCOMP

Con el comando DISKCOPY se copia el contenido de un disquete en otro dis- Normalmente utilizaremos DISKCOMP después de utilizar el comando DISKCOPY, para asegurarnos de que los dos disquetes son idénticos. Asimismo, podemos comparar sólo la primera cara de los disquetes mediante la opción correspondiente.

Hay que resaltar que este comando compara *disquetes completos.* En caso de querer comparar ficheros por separado se utiliza la orden COMP.



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼